UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

LEAST SQUARE MULTI-CLASS KERNEL MACHINES WITH PRIOR KNOWLEDGE AND APPLICATIONS

A Dissertation

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

By

OLUTAYO O. OLADUNNI
Norman, Oklahoma
2006

UMI Number: 3207058

UMI Microform 3207058

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

LEAST SQUARE MULTI-CLASS KERNEL MACHINES WITH PRIOR KNOWLEDGE AND APPLICATIONS

A DISSERTATION APPROVED FOR THE
SCHOOL OF INDUSTRIAL ENGINEERING

By

_____

Dr. T.B. Trafalis, Chair

_____

Dr. T.R. Rhoads

_____

Dr. S. Karabuk

_____

Dr. D.V. Papavassiliou

_____

Dr. S.O. Osisanya

Acknowledgements

I would like to express my thanks to the faculty and staff of the School of Industrial Engineering, The University of Oklahoma for having supported me throughout the course of my academic studies and endeavors. I also like to express my gratitude to Dr. T.B. Trafalis, Dr. T.R. Rhoads, Dr. S. Karabuk, Dr. D.V. Papavassiliou and Dr. S. Osisanya who have provided advice and counsel while serving on my committee during the course of my doctoral studies. A special thanks to Dr. T.B. Trafalis for having served as my advisor, and providing significant guidance to the development and implementation of my research plan. His expertise and guidance were of great importance for this research. I would also like to thank my parents, Otunba (Dr.) S.A. Oladunni and Olori M. Oladunni for their continued support & encouragement throughout my academic life, and towards my personal development. Finally, I would like to acknowledge the support of the National Science Foundation under NSF Grant EIA-0205628.

This dissertation is dedicated to my parents, brothers and sister.

My parents,

SOLOMON AYODELE OLADUNNI,

MODUPE OLADUNNI.

My brothers,

DAMOPE (AYODEJI) OLADUNNI,

ODUNAYO OLADUNNI,

MAYOWA OLADUNNI.

My sister,

JUMOKE OLADUNNI.

Contents

List of Tables

List of Figures

Abstract

In this study, the problem of discriminating between objects of two or more classes with (or without) prior knowledge is investigated. We present how a two-class discrimination model with or without prior knowledge can be extended to the case of multi-categorical discrimination with or without prior knowledge. The prior knowledge of interest is in the form of multiple polyhedral sets belonging to one or more categories, classes, or labels, and it is introduced as additional constraints into a classification model formulation. The solution of the knowledge-based support vector machine (KBSVM) model for two-class discrimination is characterized by a linear programming (LP) problem, and this is due to the specific norm (L_1 or L_∞) that is used to compute the distance between the two classes. We propose solutions to classification problems expressed as a single unconstrained optimization problem with (or without) prior knowledge via a regularized least square cost function in order to obtain a linear system of equations in input space and/or dual space induced by a kernel function that can be solved using matrix methods or iterative methods. Advantages of this formulation include the explicit expressions for the classification weights of the classifier(s); its ability to incorporate and handle prior knowledge directly to the classifiers; its ability to incorporate several classes in a single formulation and provide fast solutions to the optimal classification weights for multi-categorical separation.

Comparisons with other learning techniques such as the least square SVM & MSVM developed by Suykens & Vandewalle (1999b & 1999c), and the knowledge-based SVM

developed by Fung et al. (2002) indicate that the regularized least square methods are more efficient in terms of misclassification testing error and computational time.

Chapter 1

Introduction

1.1 Research Objectives

Support Vector Machines (SVMs), developed by Vapnik (1995, 1998), have been successfully applied to a wide range of problems. However, during implementation of the method, various problems arise such as selecting or estimating the appropriate tradeoff parameter in the case of inseparability, and kernel function in the case of nonlinear separability. When considering the robustness of the model, one must take the following into account: incorporating uncertainty into the model (Fung & Mangasarian, 2005), incorporating prior knowledge (Fung et al., 2002 & 2003; Mangasarian et al., 2004), and obtaining a sparse representation of classifiers (Suykens et al., 2000).

The problems mentioned are rather significant. However, the shortfalls of SVMs should not greatly disturb the solution of classification problems, they should be investigated for a more comprehensive and/or alternative solution to the same classification problem. The SVM model as originally proposed requires the construction of several binary SVM classifiers to solve a multi-class problem. Although theoretically this scheme may seem practical, it becomes increasingly tedious to continue to perform the same repeated procedure of providing a solution for all independent classification models as this tends to increase the time used to obtain all solutions. Therefore a practical problem is a generalized extension of the support vector machine model for binary classification to accommodate multi-classification problems.

The purpose of the research is as follows:

i. The development of, and solution to, a pairwise multi-classification SVM model for multi-categorical discrimination of sets involving three or more classes.

   Current research in this area targets the construction of single optimization models for the reduction of the computational effort needed to solve the resulting large scale optimization problems. Practical attempts involve solving k SVM models, where k is the number of classes and $k(k-1)/2$ is the number of SVM classifiers (Hsu & Lin, 2002; Santosa et al., 2002). Other attempts involve the solution of a single optimization problem using all data at once (Bredensteiner & Bennet, 1999; Weston & Watkins, 1999; Szedmak & Shawe-Taylor, 2004; Oladunni & Trafalis, 2005; Trafalis & Oladunni, 2005). The latter are arguably the most well constructed multi-class formulations most closely aligned with Vapnik's structural minimization principle (Vapnik, 1995 & 1998).

   Rifkin & Klautau (2004) developed a regularized least square (RLSC) model using the Tikhonov regularization (Tikhonov & Arsenin, 1977) approach. In that study, they showed that the standard SVM and RLSC model report similar empirical performance. However, from a computational point of view, the RLSC model computation time is more consistent, hence more favorable for generating classifiers. To distinguish between the proposed formulation and RLSC, note that in the intended formulation the unregularized bias offset is considered, and the intended model problem is employing a pairwise strategy (Oladunni & Trafalis, 2005) for multi-classification.

Investigations will include the development of a linearly and nonlinearly separable binary classification SVM, extended to the case of multi-categorical discrimination expressed as a single unconstrained optimization problem.

Advantages of this formulation include the explicit expressions for the classification weights of the classifier(s); its ability to incorporate several classes in a single formulation and provide fast solutions to the optimal classification weights for multi-categorical separation; and its ability to provide solutions without the use of specialized solver-software.

ii. The development of, and robust solution to, a knowledge based SVM model for pairwise multi-categorical discrimination of sets with prior knowledge.

There are circumstances where data collection can prove to be expensive or even impossible to acquire; there might only be a certain relationship belonging to a specific class, label, or category. With such relationships, the proposed model attempts to find a robust classifier which incorporates prior knowledge in the form of polyhedral sets belonging to each class. This results in a model that combines both the knowledge set and the data. Traditional SVMs are not equipped to handle such a problem because they are completely data driven. They require a given dataset which they will analyze for hidden relationships and possible decision making. Considering the need for incorporating prior knowledge and finding solutions to such models, the second objective of the dissertation is intended to provide a formulation for knowledge-based SVM classifiers using a regularization approach and extend it to the case of multi-classification (classification involving three or more classes).

1.2 Scope and Organization of Dissertation Proposal

The organization of the dissertation is as follows: In chapter 2, a review of concepts and definitions are presented. Concepts include; SVM, least square (LS) method, LS-SVM, Multiclass-SVM (MSVM) and knowledge based SVM (KB-SVM). In chapter 3, a review of the relevant literature with respect to SVM, MSVM and KB-SVM are discussed. In chapter 4, a model for linear and nonlinear multi-classification of sets is proposed. In chapter 5, a model for knowledge based linear and nonlinear multi-classification of sets with prior knowledge is proposed. Applications of the proposed model are discussed in chapter 6. Preliminary computational results are presented in chapter 7. Finally in chapter 8, summary, conclusion and future research are discussed.

Chapter 2

Concepts & Definitions

In this chapter, several concepts and definitions are presented to facilitate the research objectives of the dissertation.

2.1 Support Vector Machines

SVMs (Vapnik 1995, 1998) have been applied to a wide range of problems. Their prime advantage for classification problems is their ability to perform a mapping of the variables in a high (possibly infinite) feature space, providing an avenue for exploring nonlinear kernel-based classifiers (Burges, 1998; Cristianini & Shawe-Taylor, 2000). SVMs are equipped to discriminate between sets that are linearly separable, linearly inseparable, and nonlinearly separable. SVMs avoid overfitting by maximizing the margin between two classes of training data; i.e., maximizing the distance between the separating hyperplane and the training data on either side of it (see Figure 1).

In this section we consider the two-class classification problem. Specifically, l data points $(x_i, y_i)_{i=1}^{l}$ are given, where $x_i \in \Re^n$ are the input training vectors, and $y_i \in \{+1, -1\}$ are the corresponding labels.

2.1.1 Linearly Separable Problem

Making the assumption that the classification problem is linearly separable, the problem becomes finding a (w, γ) that defines a separating hyperplane such that the following separation constraints hold (see Figure 1):

$$\begin{aligned} w \cdot x_i - \gamma &\geq 1, \quad y_i = 1 \\ w \cdot x_i - \gamma &\leq -1, \quad y_i = -1 \end{aligned} \tag{2.1.1.1}$$

where w is the normal vector perpendicular to the separating hyperplane, and γ is the bias—the position of the hyperplane analyzed in input space. Combining constraints (2.1.1.1) into one set of inequalities gives:

$$y_i \left(w \cdot x_i - \gamma \right) \geq 1, \quad i = 1, .., l \tag{2.1.1.2}$$

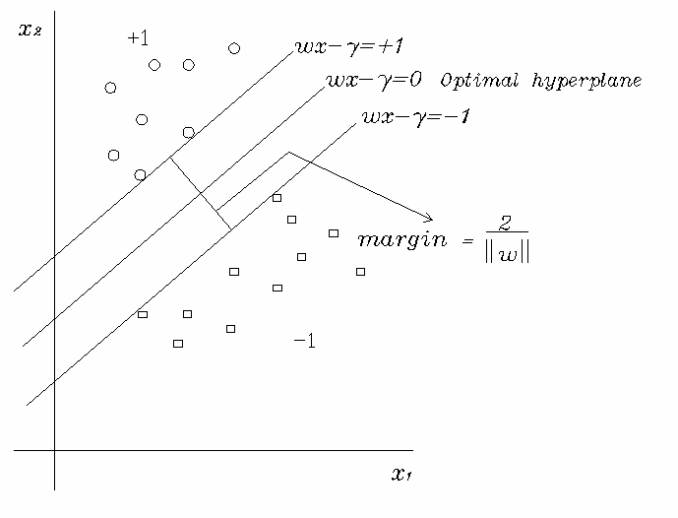

Figure 1: Linearly separable problem. The optimal hyperplane is orthogonal to the shortest line connecting the two classes, and intersects it halfway; circles in +1 class and squares in -1 class.

The formulation of a linearly separable problem written in its primal form (Burges, 1998; Cristianini & Shawe-Taylor, 2000; Chang & Lin, 2001; Hsu et al., 2003) is as follows:

$$\begin{aligned} &\min_{w,\gamma} \frac{1}{2} \|w\|^2 \\ &s.t. \ y_i \left(w \cdot x_i - \gamma \right) \geq 1, \quad i = 1, \ldots, l \end{aligned} \tag{2.1.1.3}$$

Where $\|w\|^2 = w^T w$ is the square of the 2-norm of the normal vector defining the separating hyperplane and it is a convex function.

Kuhn and Tucker (Reklaitis et al., 1983; Bazaraa et al., 1993) have developed the necessary and sufficient optimality conditions by using the concept of the Lagrangian function. For the SVM model its Lagrangian function is given by:

$$L(w,\gamma,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{l}\alpha_i\left(y_i(w\cdot x_i-\gamma)-1\right) \tag{2.1.1.4}$$

At the global saddle point, L should be minimized with respect to w, γ and maximized with respect to $\alpha_i \geq 0$, where α_i are positive Lagrange multipliers.

The Karush Kuhn-Tucker conditions (KKT) for the primal problem are given below:

$$\frac{\partial L}{\partial \gamma} = 0 \Rightarrow -\sum_{i=1}^{l}\alpha_i y_i = 0 \tag{2.1.1.5}$$

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w - \sum_{i=1}^{l}\alpha_i y_i x_i = 0 \Rightarrow w = \sum_{i=1}^{l}\alpha_i y_i x_i \tag{2.1.1.6}$$

$$\alpha_i\left[y_i\left(w\cdot x_i-\gamma\right)-1\right]=0 \tag{2.1.1.7}$$

Substituting equations (2.1.1.5) and (2.1.1.6) into (2.1.1.4), provides the Wolfe dual problem in the following form:

$$\begin{aligned}&\max_{\alpha} w(\alpha)=\max_{\alpha}\sum_{i=1}^{l}\alpha_i-\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j\left\langle x_i,x_j\right\rangle\\ &s.t.\ \sum_{i=1}^{l}\alpha_i y_i=0\\ &\quad \alpha_i\geq 0,\quad i=1,\ldots,l\end{aligned} \tag{2.1.1.8}$$

The *optimal point* solution is given by:

$$w^* = \sum_{i=1}^{l}\alpha_i^* y_i x_i\ , \tag{2.1.1.9}$$

where a training vector for which $\alpha_i^* > 0$ is called a support vector.

The decision function is:

$$f(x) = sign\left(\sum_{i=1}^{l} \alpha_i y_i \left\langle x_i, x_j \right\rangle - \gamma \right) \tag{2.1.1.10}$$

The KKT complementary condition (2.1.1.7) can be used to determine the threshold value, γ, for any i such that α_i is not zero.

2.1.2 Linearly Inseparable Problem

The problem is to find a (w, γ) that defines a separating hyperplane such that the following separation constraints hold:

$$\begin{aligned} w \cdot x_i - \gamma &\geq 1 - \xi_i, \quad y_i = 1 \\ w \cdot x_i - \gamma &\leq -1 + \xi_i, \quad y_i = -1 \end{aligned} \tag{2.1.2.1}$$

where ξ_i is a non-negative slack. Condition (2.1.2.1) is a linearly inseparable separation constraint analyzed in input space; its combination results into a single set of inequalities:

$$y_i \left(w \cdot x_i - \gamma \right) \geq 1 - \xi_i, \quad i = 1, .., l \tag{2.1.2.2}$$

The formulation of a linearly inseparable problem can be written in its primal form (Burges, 1998; Cristianini & Shawe-Taylor, 2000; Chang & Lin, 2001; Hsu et al., 2003) as follows:

$$\begin{aligned} &\min_{w,\gamma,\xi} \frac{1}{2} \|w\|^2 + \lambda \sum_{i=1}^{l} \xi_i \\ &s.t. \ y_i \left(w \cdot x_i - \gamma \right) + \xi_i \geq 1 \\ &\quad \xi_i \geq 0, \ i = 1, \ldots, l \end{aligned} \tag{2.1.2.3}$$

The linearly inseparable case has an error term, $\lambda \sum_{i=1}^{l} \xi_i$, included in the objective function of model (2.1.1.3) and it's minimized together with the square of the 2-norm. It is evident that if two disjoint sets are linearly separable, then the error slacks will be zero (see Figure 1).

The non-negative slack variable, ξ_i , measures the degree of violation of the constraints (see Figure 2). The parameter λ is a constant called the regularization parameter. It controls the tradeoff between minimizing training error and minimizing the 2-norm of the normal vector (generalization ability). This parameter is chosen by the modeler, keeping in mind that a larger λ corresponds to a higher penalty of errors with less generalization ability resulting in a more complex model. A smaller λ corresponds to fewer penalties with higher generalization ability resulting in a less complex model. Thus, there is a need to select a suitable value for λ to control the tolerance level for errors. For the quadratic programming solution, λ is usually determined by employing cross validation techniques (Hsu et al., 2003).

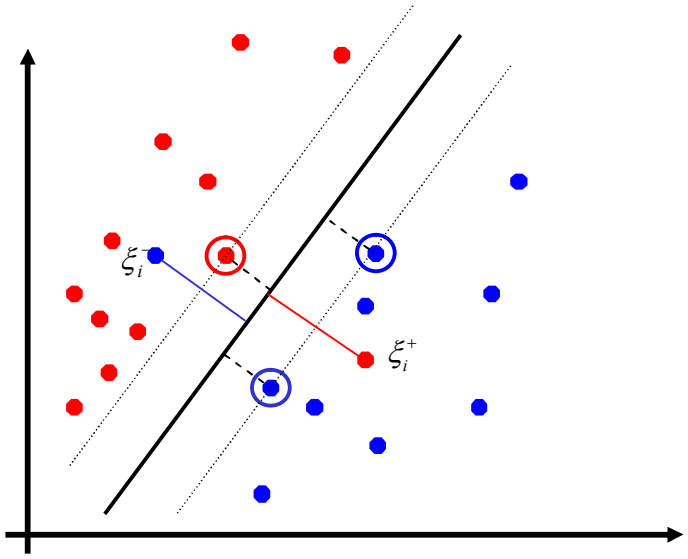

Figure 2: A Support Vector Machine classification schematic with violation of separation constraints (linearly inseparable problem).

To apply the KKT (Reklaitis et al., 1983; Bazaraa et al., 1993), the Lagrangian of problem (2.1.2.3) is defined as:

$$L(w,\gamma,\xi,\alpha,\beta)=\frac{1}{2}\|w\|^2+\lambda\sum_{i=1}^{l}\xi_i-\sum_{i=1}^{l}\alpha_i\left(y_i(w\cdot x_i-\gamma)-1+\xi_i\right)-\sum_{i=1}^{l}\beta_i\xi_i \qquad (2.1.2.4)$$

At the global saddle point, L should be minimized with respect to w, γ, ξ, and maximized with respect to $\alpha_i, \beta_i \geq 0$; where α_i, β_i are positive Lagrange multipliers.

The Karush Kuhn-Tucker conditions (KKT) for the primal problem are given below:

$$\frac{\partial L}{\partial \gamma} = 0 \Rightarrow -\sum_{i=1}^{l}\alpha_i y_i = 0 \qquad (2.1.2.5)$$

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w - \sum_{i=1}^{l}\alpha_i y_i x_i = 0 \Rightarrow w = \sum_{i=1}^{l}\alpha_i y_i x_i \qquad (2.1.2.6)$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow \lambda-\alpha_i-\beta_i = 0 \Rightarrow \alpha_i+\beta_i = \lambda \qquad (2.1.2.7)$$

$$y_i(w\cdot x_i-\gamma)-1+\xi_i \geq 0 \qquad (2.1.2.8)$$

$$\xi_i \geq 0 \tag{2.1.2.9}$$

$$\alpha_i \geq 0 \tag{2.1.2.10}$$

$$\beta_i \geq 0 \tag{2.1.2.11}$$

$$\beta_i \xi_i = 0 \tag{2.1.2.12}$$

$$\alpha_i \left[y_i \left(w \cdot x_i - \gamma \right) - 1 + \xi_i \right] = 0 \tag{2.1.2.13}$$

Equation (2.1.2.7) combined with equation (2.1.2.12) indicates that $\xi_i = 0$ if $\alpha_i < \lambda$; therefore substituting the necessary KKT (2.1.2.5) & (2.1.2.6) in the Lagrangian (2.1.2.4) provides the Wolfe dual problem below:

$$\begin{aligned} &\max_{\alpha} w(\alpha) = \max_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \left\langle x_i, x_j \right\rangle \\ &s.t \quad \sum_{i=1}^{l} \alpha_i y_i = 0 \\ &\qquad 0 \leq \alpha_i \leq \lambda, \qquad i = 1, \ldots, l \end{aligned} \tag{2.1.2.14}$$

We can use the KKT complementary conditions—(2.1.2.12) & (2.1.2.13)—to determine the threshold value, γ, for any i such that α_i is not zero.

2.1.3 Kernel Functions for Nonlinear Separable Problems

In the case of nonlinear separation, the input vector x, of the SVM, can be transformed into a higher dimensional space called the feature space, using a function $\phi : x \in X \rightarrow \phi(x) \in F$. This transformation allows the solution of the classification problem to be solved in feature space utilizing linear techniques. Note that the function $\phi(x)$ might not be available or cannot be computed. However, while $\phi(x)$ might not be available, one can still compute the inner product $\phi(x_1) \cdot \phi(x_2)$ in feature space implicitly

through a kernel function. This function can be viewed as determining the similarity or distance between the input vectors. The inner product in feature space can be expressed by the kernel function (Cristianini & Shawe-Taylor, 2000):

$$k : \Re^n \times \Re^n \to \Re : k(x_1, x_2) = \phi(x_1) \cdot \phi(x_2) \tag{2.1.3.1}$$

Figure 3 illustrates the transformation process from input space to feature space.

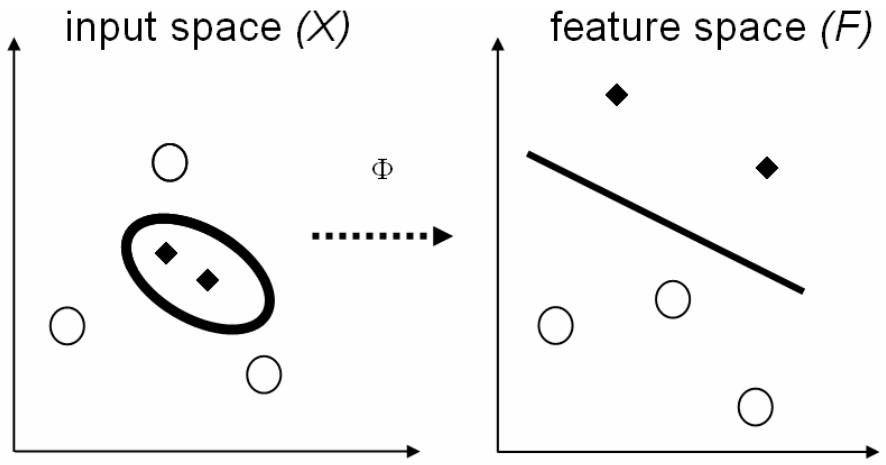

Figure 3: Feature space schematic (Schölkopf and Smola, 2002)

Therefore in the case of nonlinear separability, we can replace $\langle x_i, x \rangle$ in the dual problem (2.1.2.14), and decision function (2.1.2.10), by the kernel function $k(x_i, x)$. Then problem (2.1.2.14) becomes:

$$\begin{aligned} &\max_{\alpha} w(\alpha) = \max_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ &s.t \quad 0 \le \alpha_i \le \lambda, \quad i = 1, ..., l \\ &\quad \sum_{i=1}^{l} \alpha_i y_i = 0 \end{aligned} \tag{2.1.3.2}$$

and the decision function becomes:

$$f(x) = sign\left(\sum_{i=1}^{l} \alpha_i y_i k(x_i, x) - \gamma \right) \tag{2.1.3.3}$$

Below are examples of two widely used kernel functions:

- The polynomial kernel.

$$k(x_i, x) = (x_i^T x + 1)^P, \tag{2.1.3.4}$$

where p is the degree of the polynomial for the polynomial kernel function.

- The Gaussian radial basis kernel function (rbf)

$$k(x_i, x) = \exp\left(-\frac{\|x_i - x\|^2}{2\sigma^2}\right), \tag{2.1.3.5}$$

where σ is the spread for the Gaussian rbf kernel. There are several other kernel functions; see Cristianini & Shawe-Taylor, 2000.

2.2 Tikhonov Regularization

The main idea underlying regularization theory is that an ill-posed problem or in particular, a problem of approximating the functional relation between x and y given a finite number of l points $(x_i, y_i)_{i=1}^{l}$, can be formulated as a variational problem which contains both that data and prior smoothness information. The smoothness of the function is taken into account by defining a smoothness functional $\varphi(f)$ in such a way that smaller values of the functional correspond to smoother functions. To find a function that is simultaneously close to the data and is also smooth, leads to the approximation problem arising from the minimization of the following quadratic functional (Tikhonov & Arsenin, 1977; Ivanov, 1976):

$$H[f] = \frac{1}{l}\sum_{i=1}^{l}\left(y_i - f(x_i)\right)^2 + \lambda\varphi(f) \tag{2.2.1}$$

for a fixed λ, called the regularization parameter. The first term is an L_2 loss function for empirical risk, enforcing closeness to the data. The second term called stabilizer, enforces smoothness (generalization ability), and the regularization parameter controls the tradeoff. The minimizer of functional (2.2.1) is as follows:

$$f(x) = \sum_{i=1}^{\infty} c_i \varphi_i(x) \quad \text{or} \quad f(x) = \sum_{i=1}^{\infty} c_i \varphi_i(x) - b \tag{2.2.2}$$

For classification the decision function is *sign*[*f(x)*]. The image space of $\varphi(x)$ is called the feature space, and its dimensionality is the number of basis elements $\varphi_i(x)$ which do not necessarily have to be infinite. The choice of $\varphi(x)$ defines the feature space where the data x_i are mapped. In terms of classification, the smoothness functional together with a loss function can be geometrically interpreted in terms of the margin.

2.3 Least Square Method

The Least Square (LS) approach provides an approximate solution to a uniquely determined, over-determined and under-determined linear system of equations. Rather than solving the equations exactly we attempt to minimize the sum of square errors. Computational techniques for solving least square problems include decomposition methods such as orthogonal matrix factorization, singular value decomposition, Cholesky decomposition, etc. Iterative methods such as Newton's method, steepest descent method, conjugate gradient method, etc. are also used (Lewis, et al., 2006).

2.3.1 Over-Determined Linear System

The least square problem for the over-determined case (m > n) is:

$$\min f(x_{ls}) = \left\| b - Ax_{ls} \right\|^2 \tag{2.3.1.1}$$

where $A \in R^{m \times n}$ is a matrix, $b \in R^m$ is a vector and $x_{ls} \in R^n$ is a vector.

To minimize $f(x_{ls})$, the gradient and Hessian of $f(x_{ls})$ are given by:

$$\nabla f(x_{ls}) = -2A^T b + 2(A^T A)x_{ls} \tag{2.3.1.2}$$

and

$$\nabla^2 f(x_{ls}) = 2(A^T A) \tag{2.3.1.3}$$

Applying the first order conditions by setting the gradient in (2.3.1.2) equal to zero, provides a minimizing solution, x_{ls}, to the linear system of simultaneous equations known as *normal equations*:

$$(A^T A)x_{ls} = A^T b \tag{2.3.1.4}$$

If $\nabla^2 f(x_{ls})$ is a positive definite (PD) matrix, then we get the minimizing solution of $f(x_{ls})$,

$$x_{ls} = (A^T A)^{-1} A^T b \tag{2.3.1.5}$$

2.3.2 Under-Determined Linear System

The least square problem for the under-determined case (m < n) is:

$$\begin{aligned} &\min f(x_{ls}) = \left\| x_{ls} \right\|^2 \\ &\text{s.t.} \quad Ax_{ls} = b \end{aligned} \tag{2.3.2.1}$$

where $A \in R^{m \times n}$ is a matrix, and $b \in R^m$ is a vector.

Using the concept of the Lagrangian multipliers, $\lambda_{ls} \in R^m$, problem (2.3.2.1) is expressed as an unconstrained optimization problem of the form (Lagrangian):

$$L(\lambda_{ls}, x_{ls}) = \|x_{ls}\|^2 + \lambda_{ls}{}^T (b - Ax_{ls}) = x_{ls}{}^T x_{ls} + \lambda_{ls}{}^T (b - Ax_{ls}) \quad (2.3.2.2)$$

First order optimality conditions for $L(\lambda_{ls}, x_{ls})$:

$$\nabla_{x_{ls}} L(\lambda_{ls}, x_{ls}) = 2x_{ls} - A^T \lambda_{ls} = 0 \quad (2.3.2.3)$$

$$\nabla_{\lambda_{ls}} L(\lambda_{ls}, x_{ls}) = b - Ax_{ls} = 0 \quad (2.3.2.4)$$

Rearranging equations (2.3.2.3) and (2.3.2.4), it follows that $x_{ls} = \frac{1}{2} A^T \lambda_{ls}$ and

$b = \frac{1}{2}(AA^T)\lambda_{ls}$ respectively. Assuming that $AA^T \in R^{m \times m}$ is invertible,

then $\lambda_{ls} = 2(AA^T)^{-1} b$. Substituting λ_{ls} into equation (2.3.2.3) yields the minimizing solution given below:

$$x_{ls}* = A^T (AA^T)^{-1} b \quad (2.3.2.5)$$

2.3.3 Uniquely Determined Linear System

The least square problem for the uniquely determined case (m = n):

Assuming the columns of A are linearly independent and span space R^n ; then the vector b can be expressed as a linear combination of columns of A.

With linear independence, the solution to system (2.3.3.1) below,

$$Ax_{ls} - b = 0 \quad (2.3.3.1)$$

is given by:

$$x_{ls}* = A^{-1} b \quad (2.3.3.2)$$

2.3.4 Unified Approach

In the event that $A^T A$ or AA^T are not invertible matrices, a regularized approach (Tikhonov regularization, 1977), which expresses the system of equations as a least square, unconstrained, minimization problem of the form below is used (Lewis, et al., 2006):

$$\min f(x_{ls}) = \frac{\alpha_{ls}}{2} x_{ls}{}^T x_{ls} + \frac{1}{2}\left\|(b - Ax_{ls})\right\|^2 \tag{2.3.4.1}$$

α_{ls} is the tradeoff constant that seeks to find a compromise between obtaining both the minimum norm solution and minimum residual solution. This form is especially useful for rank deficient/singular matrices and badly conditioned Hessians. In the case that the Hessian is not positive definite (PD) we force the matrix or Hessian to become positive definite for a suitable $\alpha_{ls} > 0$, and determine a solution, x, that minimizes $f(x)$.

$$f(x_{ls}) = \frac{\alpha_{ls}}{2} x_{ls}{}^T x_{ls} + \frac{1}{2}\left(b^T b - 2b^T A x_{ls} + x_{ls}{}^T A^T A x_{ls}\right) \tag{2.3.4.2}$$

The optimality conditions for $f(x_{ls})$ are as follows:

First order conditions

$$\nabla f(x_{ls}) = \alpha_{ls} I x_{ls} - A^T b + A^T A x_{ls} = 0 \tag{2.3.4.3}$$

Second order conditions

$$\nabla^2 f(x_{ls}) = A^T A + \alpha_{ls} I\,, \tag{2.3.4.4}$$

where $\nabla^2 f(x_{ls})$ is PD

The minimum solution is obtained from the first order relation for the m > n case.

$$\begin{aligned} &A^T A x_{ls} + \alpha_{ls} I x_{ls} = A^T b \;\Rightarrow\; (A^T A + \alpha_{ls} I) x_{ls} = A^T b \\ &\Rightarrow x_{ls}{}^* = (A^T A + \alpha_{ls} I)^{-1} A^T b \end{aligned} \tag{2.3.4.5}$$

For m < n case, the matrix identity is used (Lewis, et al., 2006).

$$\begin{aligned}(A^T A+\alpha_{ls} I)^{-1} A^T &= \alpha_{ls}^{-1} I A^T (I+\alpha_{ls}^{-1} A A^T)^{-1} \\ &= \alpha_{ls}^{-1} A^T [\alpha_{ls}^{-1}(\alpha_{ls} I + A A^T)]^{-1} \\ &= A^T [\alpha_{ls} I + A A^T]^{-1}\end{aligned} \tag{2.3.4.6}$$

Substituting (2.3.4.6) into the RHS of solution (2.3.4.5) becomes:

$$x_{ls}^* = A^T (\alpha_{ls} I + A A^T)^{-1} b \tag{2.3.4.7}$$

The least square approach is very sensitive to the structure and properties of its matrix. In order to guarantee a global minimizer it is essential that the matrix be positive definite.

2.4 Least Square Support Vector Machines

Consider the two-class classification problem. Specifically, l data points $(x_i, y_i)_{i=1}^{l}$ are given, $x_i \in \Re^n$ are the input training vectors, and $y_i \in \{+1, -1\}$ are the corresponding labels. The least square version to the SVM classifier is given by the following optimization problem (Suykens &Vandewalle, 1999b):

$$\begin{aligned}&\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + \lambda \frac{1}{2}\sum_{i=1}^{l} \xi_i^2 \\ &s.t.\ \ y_i\left(w \cdot \phi(x_i) + \gamma\right) = 1 - \xi_i, \quad i = 1, \ldots, l\end{aligned} \tag{2.4.1}$$

The mapping function, $\phi: x \in X \to \phi(x) \in F$, maps the input space to a high dimensional feature space. This formulation has equality constraints rather than inequality constraints, and takes into account a penalized sum of square error term.

Similar to Vapnik's SVM, its Lagrangian function is defined below:

$$L(w, \gamma, \xi, \alpha) = \frac{1}{2}\|w\|^2 + \lambda \sum_{i=1}^{l} \xi_i^2 - \sum_{i=1}^{l} \alpha_i \left(y_i (w \cdot \phi(x_i) + \gamma) - 1 + \xi_i\right) \tag{2.4.2}$$

Where α_i are the Lagrange multipliers and can either be positive or negative due to the equality constraints of problem (2.4.1). From the conditions for optimality, the KKT system is obtained.

$$\frac{\partial L}{\partial \gamma} = 0 \;\Rightarrow\; -\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{2.4.3}$$

$$\frac{\partial L}{\partial w} = 0 \;\Rightarrow\; w - \sum_{i=1}^{l} \alpha_i y_i \phi(x_i) = 0 \;\Rightarrow\; w = \sum_{i=1}^{l} \alpha_i y_i \phi(x_i) \tag{2.4.4}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \;\Rightarrow\; \lambda \xi_i - \alpha_i = 0 \;\Rightarrow\; \alpha_i = \lambda \xi_i \tag{2.4.5}$$

$$\frac{\partial L}{\partial \alpha_i} = 0 \;\Rightarrow\; y_i (w \cdot \phi(x_i) + \gamma) - 1 + \xi_i = 0 \tag{2.4.6}$$

Eliminating variables *w and* ξ , the following linear system is obtained,

$$\begin{pmatrix} 0 & y^T \\ y & ZZ^T + \lambda^{-1} I \end{pmatrix} \begin{pmatrix} \gamma \\ \alpha \end{pmatrix} = \begin{pmatrix} 0 \\ e \end{pmatrix} \tag{2.4.7}$$

where $y = [y_1, ..., y_l]$, $\xi = [\xi_1, ..., \xi_l]$, $\alpha = [\alpha_1, ..., \alpha_l]$ and *e* is the vector of ones with appropriate dimension. To guarantee a solution, Mercer's condition (Burges, 1998; Cristianini & Shawe-Taylor, 2000) can be applied to matrix $\Omega = ZZ^T$.

$$\Omega_{ij} = y_i y_j \phi(x_i) \cdot \phi(x_j) = y_i y_j K(x_i, x_j) \tag{2.4.8}$$

The decision function becomes

$$f(x) = sign\left(\sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + \gamma \right) \tag{2.4.9}$$

The solution to system (2.4.7) provides the classification weights in dual space, and when used as parameters in the decision function, a new point can be classified.

2.5 Multi-classification Support Vector Machines

In this section, the most general formulations that address the discrimination of two or more classes ($k \geq 2$) are presented.

2.5.1 One-Against-All (OAA) Method

In the OAA method, *k* SVM classifiers are constructed where *k* is the number of classes. Each model is trained on data for one class versus the rest of the classes. For example, if we have a three-class classification problem, we have to construct 3 classifiers, P^{1v2+3}, P^{2v1+3}, P^{3v1+2}. When we train P^{1v2+3}, all points from class 1 are labeled with +1 and all the points from class 2 and 3 are labeled with -1. Similarly for P^{2v1+3}, class 2 is labeled with +1, class 1 and 3 with -1, and for P^{3v1+2}, class 3 has label +1, and class 1 and 2 are labeled -1. Given l data points $(x_i, y_i)_{i=1}^{l}$; $x_i \in \Re^n$ are the input training vectors, and $y_i \in \{1,....,k\}$ is the class of x_i . The *i*-th SVM model solves the following problem of the form:

$$
\begin{aligned}
&\min_{w^i,\gamma^i,\xi_t^i} \ \frac{1}{2}\left\|w^i\right\|^2 + \lambda\sum_{t=1}^{l}\xi_t^i \\
&s.t. \\
&(w^i)^T\phi(x_t) + \gamma^i \geq 1 - \xi_t^i, \quad if\ y_t = i \\
&(w^i)^T\phi(x_t) + \gamma^i \leq -1 + \xi_t^i, \quad if\ y_t \neq i \\
&\xi_t^i \geq 0, \qquad t = 1,.....,l
\end{aligned}
\tag{2.5.1.1}
$$

Solving (2.5.1.1) produces *k* SVM classifiers defined by the following functions:

$$
\begin{aligned}
&(w^1)^T\phi(x) + \gamma^1, \\
&\quad\vdots \\
&(w^k)^T\phi(x) + \gamma^k,
\end{aligned}
\tag{2.5.1.2}
$$

Then we predict x as being in the class with the largest value of the decision function:

$$\text{class of } x \equiv \arg\max_{i=1,\ldots,k}[(w^i)^T \phi(x) + \gamma^i] \tag{2.5.1.3}$$

Basically we solve the dual of (2.5.1.1), whose number of variables is the same as the number of data of the problem. Hence, if we are to solve a problem with *l* data points, we then have to solve *k* quadratic programming problems where each of them has *l* data points (Hsu & Lin, 2002).

2.5.2 One-Against-One (OAO) Method

In the OAO method, one constructs *k*(*k*-1)/2 SVM classifiers, each one of which is trained on data for two classes. For example, if we have a three-class classification problem, we have to construct 3 classifiers P^{12}, P^{13}, P^{23}. When we train P^{12} all points from class 1 are labeled with +1 and all the points from class 2 are labeled with -1. Similarly for P^{13}, class 1 is labeled with +1, class 3 with -1, and for P^{23}, class 2 has label +1, and class 3 label -1. For training data from the *i*-th and the *j*-th classes, we solve the following binary classification problem (Hsu & Lin, 2002; Santosa et al., 2002):

$$\begin{aligned}
&\min_{w^{ij},\gamma^{ij},\xi_t^{ij}} \ \frac{1}{2}\left\|w^{ij}\right\|^2 + \lambda\sum_{t=1}^{l}\xi_t^{ij} \\
&s.t. \\
&(w^{ij})^T\phi(x_t) + \gamma^{ij} \geq 1 - \xi_t^{ij}, \quad if\ y_t = i \\
&(w^{ij})^T\phi(x_t) + \gamma^{ij} \leq -1 + \xi_t^{ij}, \quad if\ y_t = j \\
&\xi_t^{ij} \geq 0
\end{aligned} \tag{2.5.2.1}$$

There are different strategies for performing the testing for points not in the training set after all *k*(*k*-1)/2 classifiers have been constructed. The strategy employed is called "Max Wins" strategy. This strategy is a voting approach (Hsu & Lin, 2002; Santosa et al.,

2002). For example, if sign $[(w^{ij})^T \phi(x) + \gamma^{ij}]$ indicates that x is in the *i*-th class, then the vote for the *i*-th class is increased by one. Otherwise, the vote for the *j*-th is increased by one. Then we predict x as being in the class with the largest vote. In the case that those two classes have identical votes, select the one with the smallest index. Basically we solve the dual of (2.5.2.1), of which the number of variables is the same as the number of data points in the two classes. Hence, if each class has *l/k* data points, we have to solve *k*(*k*-1)/2 quadratic programming problems where each of them has *2l/k* data points (Hsu & Lin, 2002).

2.5.3 Pairwise Multi-classification Support Vector Machines

In trying to solve the classification problem as a single optimization problem, problem (2.5.2.1) can be generalized with the introduction of indices provided that the size of each class are of moderate size and/or it's designs are balanced (equal sizes) (Trafalis & Oladunni, 2005) (see Figure 4).

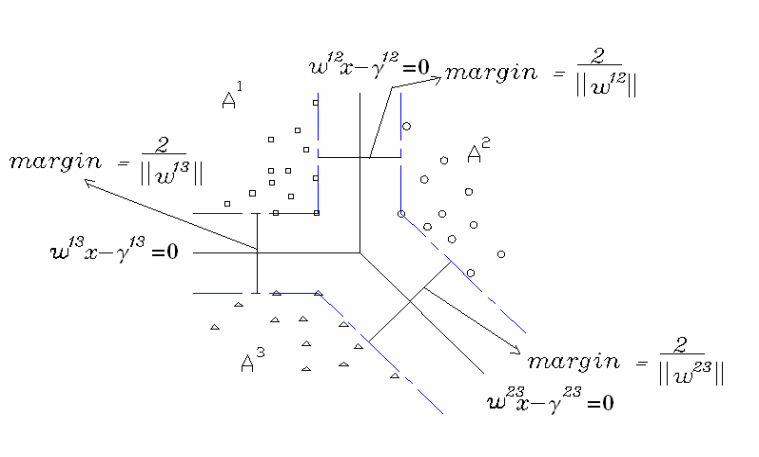

Figure 4: A Multi-Classification diagram. Three classes, class 1, 2 and 3 are represented by small squares, circles and triangles respectively; blue lines are the supporting hyperplanes for each pairwise comparison; the optimal hyperplane is orthogonal to the shortest line connecting each pairwise comparison, and intersects it halfway.

Given that the data sets in R^n are represented by a matrix $A^i \in R^{m_i \times n}$, where $i = 1,..,k$ (k classes). Let A^i be a $m_i \times n$ matrix whose rows are points in the i^{th} class, and let A^j be a $m_j \times n$ matrix whose rows are points in the j^{th} class. If $x \in R^n$ can be classified as follows:

$$\begin{aligned} & x^T w^{ij} - \gamma^{ij} \geq 1, \quad x \in A^i \\ & x^T w^{ij} - \gamma^{ij} \leq -1, \quad x \in A^j, \quad i < j \end{aligned} \tag{2.5.3.1}$$

Then below is a general approach for obtaining $k(k\text{-}1)/2$ classifiers from a single optimization model in primal form that solves problems of multi-class discrimination (Trafalis & Oladunni, 2005):

$$\begin{aligned} & \min_{w,\gamma} \ \frac{1}{2} \sum_{i<j}^{k} \left\| w^{ij} \right\|^2 + \lambda \sum_{i<j}^{k} \sum_{t=1}^{l_{ij}} \xi_t^{ij} \\ & s.t. \ y^{ij} (A^{ij} w^{ij} - \gamma^{ij}) + \xi_t^{ij} \geq 1, \quad i < j \\ & \xi_t^{ij} \geq 0 \end{aligned} \tag{2.5.3.2}$$

with $A^{ij} = \begin{bmatrix} A^i \\ A^j \end{bmatrix}$ ($m_{ij} \times n$ matrix) and $y^{ij} = \pm 1$ for classes i^{th} and j^{th} respectively. Note that $m_{ij} = m_i + m_j$. The parameter λ is a constant called the regularization parameter which controls the tradeoff between minimizing training error ξ^{ij} and minimizing the norm of the normal vector (generalization ability). The dual of problem (2.5.3.2) is given below:

$$\begin{aligned} & \max_{\alpha} \ \sum_{i<j}^{k} \sum_{c=1}^{m_{ij}} \alpha_c^{ij} - \frac{1}{2} \sum_{i<j}^{k} \sum_{c,d=1}^{m_{ij}} \alpha_c^{ij} \alpha_d^{ij} y_c^{ij} y_d^{ij} A_c^{ij}, A_d^{ij\,T} \\ & s.t. \ \sum_{c=1}^{m_{ij}} \alpha_c^{ij} y_c^{ij} = 0 \\ & \quad 0 \leq \alpha_c^{ij} \leq \lambda, \ i < j \end{aligned} \tag{2.5.3.3}$$

where $\alpha^{ij} \geq 0$ are Lagrangian multipliers, and $\alpha^{ij} > 0$ are support vectors.

2.6 Knowledge-Based Support Vector Machines

Knowledge based SVM (KBSVM) can be considered a robust method which deals with the incorporation of prior knowledge into a linear support vector machine classifier. This prior knowledge is in the form of multiple polyhedral sets, which are unique to their respective classes (see Figure 5).

Consider the problem of classifying m points in the n-dimensional input space R^n, represented by a $m \times n$ matrix, A. The membership of each point in A to a positive or negative class is specified by a $m \times m$ matrix, D, whose diagonals are labels $D_{ii} \in \{+1, -1\}$. The following knowledge sets are also given (see Figure 5):

k sets: $\{x \mid B^i x \leq b^i\}, i = 1,...,k$, belonging to A^+ (positive class)

l sets: $\{x \mid C^i x \leq c^i\}, i = 1,...,m$, belonging to A^- (negative class) (2.6.1)

The knowledge based support vector machine (KBSVM) model (Fung et al., 2002) is given as:

$$\begin{aligned}
&\min_{w,\gamma,\xi,u^i,r^i,p^i,v^j,s^j,\sigma^j} \ \|w\|_1 + \lambda e^T \xi + \mu\left(\sum_{i=1}^{k}(r^i + \rho^i) + \sum_{j=1}^{l}(s^j + \sigma^j)\right) \\
&s.t.\ \ D(Aw - e\gamma) + \xi \geq e \\
&-r^i \leq B^{iT}u^i + w \ \leq r^i \\
&\quad b^{iT}u^i + \gamma + 1 \leq \rho^i, \quad i = 1,...,k \\
&-s^j \leq C^{jT}v^j - w \ \leq s^j \\
&\quad c^{jT}v^j - \gamma + 1 \leq \sigma^j \quad j = 1,...,l \\
&\quad \xi, u^i, r^i, p^i, v^j, s^j, \sigma^j \geq 0
\end{aligned} \tag{2.6.2}$$

Where $r^i, \rho^i, i = 1,....,k$, $s^j, \sigma^j, j = 1,....,l$ are error variables that are forced to zero if linearly separable.

The solution to (2.6.2) provides the classification weights based on data and prior knowledge of the problem. Decision function ($sign\left[w^T x-\gamma\right]$) classifies a new point coming to the system.

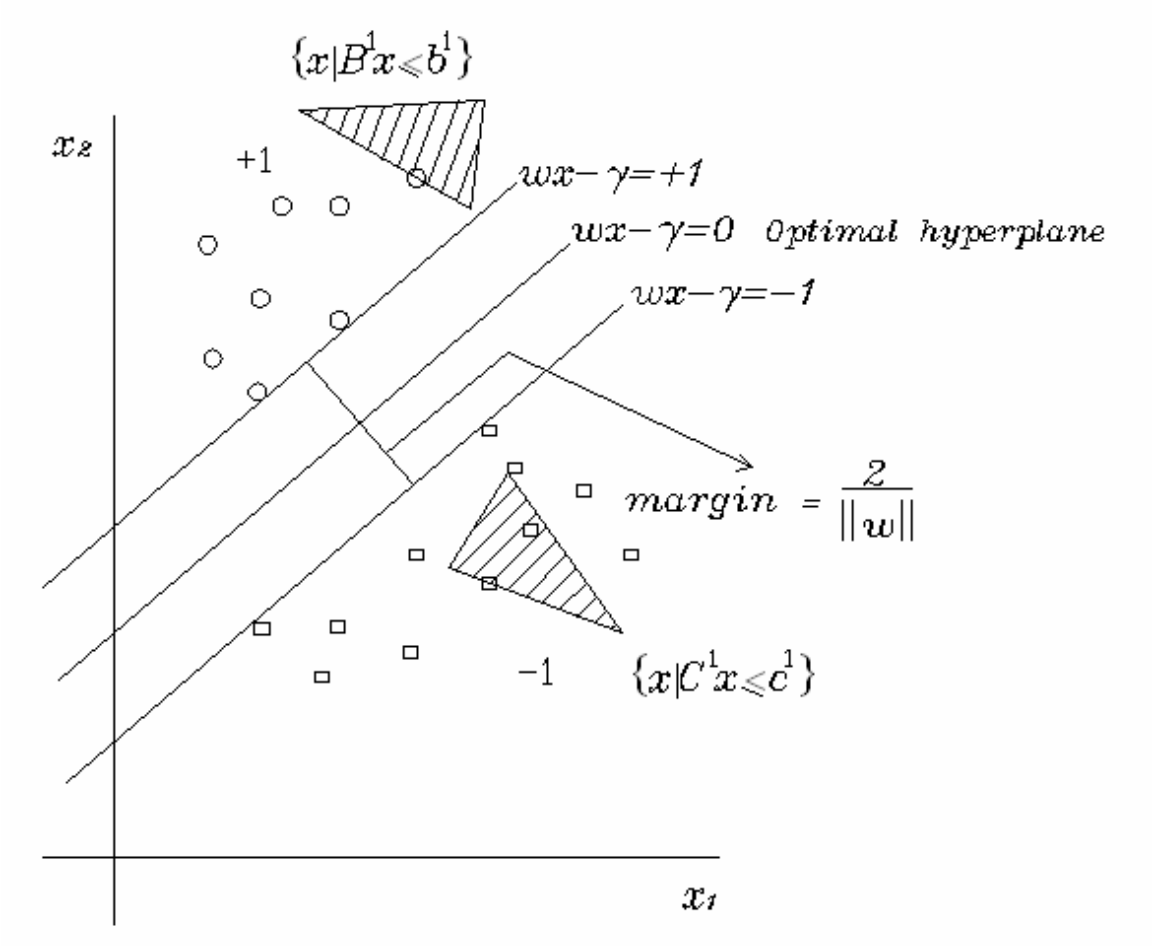

Figure 5: A knowledge-based SVM diagram, with two knowledge sets: belonging in the halfspace of class (+1), and belonging in the halfspace of class (-1).

Chapter 3

Literature Review

There have been several proposed methods for solving multi-classification problems in the support vector machine literature. In this chapter a review of the relevant SVM literature is presented.

3.1 Support Vector Machines

Vapnik's SVM (1995, 1998) was developed to solve linear and nonlinear classification problems (Burges, 1998; Cristianini & Shawe-Taylor, 2000; Mangasarian, 2000). It expresses a classification problem in quadratic programming (QP) form and guarantees a global and unique solution. The QP formulation allows the use of kernel approximations for nonlinear separability provided that Mercer's condition (Burges, 1998; Cristianini & Shawe-Taylor, 2000) isn't violated.

Suykens & Vandewalle (1999b) proposed a modified version of the SVM called the least square SVMs (LS-SVMs). In their formulation equality separation constraints were used and the square norm of the error term is minimized. This idea is similar to the ridge regression concept (Saunders et al., 1998) and Tikhonov scheme (Tikhonov, 1977), but with binary targets {-1, 1} as outputs. As a result, the classification problem becomes a linear system of equations solved using the method of least squares. In the classical QP formulation many support vector values are zero, while in the least square case the support vector values are proportional to the errors at the data points. Hence, sparsity of the solution is lost. In the case of large scale classification problems, Suykens et al.

(1999) suggested an iterative training algorithm for LS-SVMs that is based on a conjugate gradient method. The iterative procedure is considered in order to avoid storage of the input matrix.

To demonstrate the performance of the LS-SVM, Baesens et al. (2000) and Van Gestel et al. (2004) performed an empirical study of the LS-SVM on several benchmark data sets using some kernel functions. The linear kernel was used for linear separability. To attain further insight into the degree of nonlinearity of the problem a polynomial and a RBF kernel was used for nonlinear separability. Due to standard cross validation techniques for hyperparameter selection, both studies confirm that the SVM and LS-SVM with the RBF kernel achieve comparable performances and consistently yield the best results for each data set.

A drawback of the LS-SVM is that sparsity of solution is lost, and this becomes important in finding equivalence between sparse approximation and support vector machines (Girosi, 1998). To obtain a sparse solution, Suykens et al. (2000) proposed a heuristic method for pruning the support value continuum. This is done by gradually removing the least important data from the training set and re-estimating the LS-SVM. A small amount of points—e.g. 5% of training set—with smallest values in the sorted support value continuum is removed. The procedure is performed as a second stage operation for obtaining a solution to the LS-SVM classification problem.

Fung & Mangasarian (2001) developed a linear and nonlinear classification algorithm called proximal support vector machine (PSVM). The idea is to classify new points by assigning them to the closer of the two parallel planes that are as far apart as possible. Their formulation has similar interpretation to the regularized LS-SVM (Pelckmans et al.,

2004) because it depends on the strong convexity of the objective function. However, it does differ slightly from the broad perspective of regularized networks (Evgeniou et al, 2000) which is not always convex. The solution to the PSVM models is based on a linear system of equations, as opposed to the traditional SVM, that solves a quadratic programming problem requiring a considerably longer computation time.

A survey of recent developments in the context of SVMs was described in Mangasarian (2001). Four SVM algorithms were investigated; generalized SVMs (a very general mathematical programming framework for SVMs), smooth SVMs (a smooth nonlinear equation representation of SVMs solvable by a fast Newton method), Lagrangian SVMs (an unconstrained Lagrangian representation of SVMs leading to an extremely simple iterative scheme capable of solving classification problems with millions of points), and reduced SVMs (a rectangular kernel classifier that utilizes as little as 1% of the data).

Similar to the PSVM formulation, Pelckmans et al. (2004) compared three related regularization schemes for kernel machines using a *least square criterion*. The regularization schemes considered are as follows: Tikhonov & Ivanov regularization (1977 & 1976), and Morozov's (1984) discrepancy principle. Below are the cost functions of the three regularization schemes.

- Tikhonov, similar to the LS-SVMs (Suykens & Vandewalle, 1999b), where λ expresses the tradeoff between data fitting and smoothness in the trust region of parameters and noise level respectively:

$$\min_{w,b,\xi} f_T(w,\xi) = \frac{1}{2} w^T w + \frac{\lambda}{2} \sum_{i=1}^{l} \xi_i^2 \quad s.t.\ w \cdot \phi(x_i) + \gamma + \xi_i = y_i, \quad i = 1,\ldots,l \qquad (3.1.1)$$

- Morozov's discrepancy principle (Morozov, 1984), where the minimal 2-norm of w realizing a fixed noise level σ^2 is to be found:

$$\min_{w,b,\xi} f_M(w) = \frac{1}{2} w^T w \quad s.t \begin{cases} w \cdot \phi(x_i) + \gamma + \xi_i = y_i, & i = 1,\ldots,l \\ l\sigma^2 = \sum_{i=1}^{l} \xi_i^2 \end{cases} \tag{3.1.2}$$

- Ivanov's regularization (Ivanov, 1976) amounts to solving for the best fit with a 2-norm on w smaller than π^2:

$$\min_{w,b,\xi} f_I(e) = \frac{1}{2} \xi^T \xi \quad s.t \begin{cases} w \cdot \phi(x_i) + \gamma + \xi_i = y_i, & i = 1,\ldots,l \\ \pi^2 = w^T w \end{cases} \tag{3.1.3}$$

The derivative of the cost functions (3.1.1) to (3.1.3) results in a linear system which after some simple manipulation, can be solved using linear techniques. The three regularization schemes allow incorporation of prior or model-free estimates of the noise variance for tuning the regularization constant in LS-SVMs.

3.2 Multi-classification Support Vector Machines

Most developed classification models are for discriminating between two classes. To address the problem of multi-classification, researchers have adopted methods which involve solving *k* SVM models (OAA method) to produce *k* classifiers, or solving *k(k-1)/2* SVM models (OAO method) to produce *k(k-1)/2* classifiers; *k* is the number of classes. Hsu & Lin (2002) developed a decomposition strategy and made a comparison of the above methods. Methods include the OAA, OAO, and Direct Acyclic Graph SVM (DAGSVM) (Platt et al., 2000). It was reported that the OAO method and DAGSVM are more suitable for practical use, and that for large scale problems, methods that consider all data at once, in general, use fewer support vectors.

In expressing and solving a multi-class problem as a single optimization problem the following models were suggested (Bredensteiner & Bennet, 1999; Weston & Watkins,

1999; Szedmak & Shawe-Taylor, 2004; Trafalis & Oladunni, 2005; Oladunni & Trafalis, 2005). These models were developed as a generalization of the binary classification model.

Bredensteiner & Bennet (1999) developed a piecewise quadratic programming multi-classification model. A set of points $A^i, i = 1, \cdots, k$ represented by matrices $A^i \in R^{m_i \times n}$ are piecewise-linearly separable if there exist $w^i \in R^n$ and $\gamma^i \in R$ such that:

$$A^i w^i - \gamma^i e > A^i w^j - \gamma^j e, \;\; i, j = 1, ..., k, \;\; i \neq j \tag{3.2.1}$$

In canonical form

$$x^T (w^i - w^j) - e(\gamma^i - \gamma^j) \geq 1, \;\; i, j = 1, ..., k, \;\; i \neq j \tag{3.2.2}$$

The bounding plane separating classes *i* and *j* is defined as $x^T (w^i - w^j) = e(\gamma^i - \gamma^j)$.

The piecewise M-SVM problem is to minimize the following:

$$\begin{aligned} \min_{w,\gamma,\xi} \;\; & \frac{1}{2} \sum_{i<j}^{k} \left\| w^i - w^j \right\|_2^2 + \frac{1}{2} \sum_{i=1}^{k} \left\| w^i \right\|_2^2 + \lambda \sum_{i<j}^{k} \sum_{m=1}^{l} \xi_m^{ij} \\ s.t. \;\; & A^i (w^i - w^j) - e(\gamma^i - \gamma^j) \geq e - \xi_m^{ij}, \\ & \xi_m^{ij} \geq 0 \\ & i, j = 1, \cdots, k \quad i \neq j \end{aligned} \tag{3.2.3}$$

Weston & Watkins (1999) also developed a quadratic programming model that satisfies condition (3.2.1). This model is similar to problem (3.2.3), with the notable difference being the objective function. The model minimizes the following problem.

$$\begin{aligned} \min_{w,\gamma,\xi} \;\; & \frac{1}{2} \sum_{m=1}^{k} {w_m}^T w_m + \lambda \sum_{i=1}^{l} \sum_{m \neq y_i} \xi_i^m \\ s.t. \;\; & w_{y_i}^T x_i + \gamma_{y_i} \geq w_m^T x_i + \gamma_m + 2 - \xi_i^m, \\ & \xi_i^m \geq 0, \;\; i = 1, \cdots, k \;\; m \in \{1, ..., k\} \setminus y_i \end{aligned} \tag{3.2.4}$$

Both model problems (3.2.3 & 3.2.4) use the same decision function of the form:

$$f(x) = \arg\max_{k}[w_i^T x + \gamma_i], \ \ i = 1,...,k \tag{3.2.5}$$

They do not generalize better than the OAA and OAO methods. However, they do report fewer support vectors. Hence, there is no theoretical justification of the better generalization of OAA and OAO methods, which happens to be the two most widely used multi-classification SVMs techniques.

Szedmak et al. (2004) proposed a multi-class model for large sample sizes and number of features. In their formulation, the OAA framework was used, and the L_1 norm of the normal vector *w* of the separating hyperplanes is minimized. Their formulation solves *k* SVM optimization problems simultaneously, and since there are no interactions between the variables of the *k* SVM problems, their method, which essentially is the OAA considering all data at once, produces the same solution as that of the separated *k* SVM problems using the L_1 norm. The formulation is a generalization of problem (2.5.1.1) with additional L_1 norms in the objective function, and the constraints contain all data points. Considering all data at once transforms the problem into a large scale problem which can get even larger with additional data points, hence the size of the problem can be a drawback of the method, if it gets too large.

Trafalis & Oladunni (2005), also proposed a multi-classification model, but using the OAO framework. The formulation is given in problem (2.5.3.2). It solves *k(k-1)/2* SVM simultaneously, and since there are no interactions between the variables of the *k(k-1)/2* SVM problems the method can be considered a pairwise multi-classification method considering all data at once. The formulation produces the same solution as that of the separated *k(k-1)/2* SVM problems. The method works well but has a drawback related to the size of the multi-class problem. Large scale problems become very expensive to

compute the solution especially when considering a quadratic programming solution. Solving a quadratic programming problem involves the number of the data points, so an increase in data points increases the dimensionality of problem, hence requiring more time to obtain a solution.

Linear programming formulations have also been considered. Bennett & Mangasarian (1994), proposed minimizing the sum of errors violating condition (3.2.1) i.e. finding a feasible solution. Weston & Watkins, (1999) also developed a linear machine for multi-class classification by writing a dual representation problem (3.2.4). They represented the normal vector *w* in its dual representation form and minimized the dual variable α_i in the objective function. The dual representation of *w* is as follows:

$$w = \sum_{i=1}^{l} \alpha_i x_i \tag{3.2.6}$$

Recently, multi-classification problems have been investigated in the context of the least square approximations (Suykens & Vandewalle, 1999c; Oladunni & Trafalis, 2005) and the broad perspective of regularized networks (Evgeniou et al, 2000).

Suykens & Vandewalle, (1999c) extended their LS-SVMs methodology to accommodate multi-classification LS-SVM problems. The cost function and constraints (with equality constraint) of the multi-class LS-SVM is given as:

$$\begin{aligned}
&\min_{w_i,\gamma_i,\xi_{k,i}} f_{LS}^{L}(w_i,\gamma_i,\xi_{k,i}) = \frac{1}{2}\sum_{i=1}^{l} {w_i}^T w_i + \frac{\lambda}{2}\sum_{k=1}^{N}\sum_{i=1}^{L}\xi_{k,i}^2 \\
&s.t.\ \ {y}^{(1)}_{k}[{w_1}^T\phi_1(x_k)+\gamma_1] = 1-\xi_{k,1}, \quad k=1,\ldots,N \\
&\qquad {y}^{(2)}_{k}[{w_2}^T\phi_2(x_k)+\gamma_2] = 1-\xi_{k,2}, \quad k=1,\ldots,N \\
&\cdots \\
&\qquad {y}^{(L)}_{k}[{w_L}^T\phi_L(x_k)+\gamma_L] = 1-\xi_{k,L}, \quad k=1,\ldots,N
\end{aligned} \tag{3.2.7}$$

The linear system that is obtained from the above formulation after writing its Lagrangian function and taking its derivatives, becomes the classification problem to solve and is given as:

$$\begin{pmatrix} 0 & Y_L^T \\ Y_L^T & \Omega_L \end{pmatrix} \begin{pmatrix} \gamma_L \\ \alpha_L \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{3.2.8}$$

$$Y_L = \text{blockdiag}\{\begin{bmatrix} y_1^{(1)} \\ \vdots \\ y_N^{(1)} \end{bmatrix}, ..., \begin{bmatrix} y_1^{(L)} \\ \vdots \\ y_N^{(L)} \end{bmatrix}\} \tag{3.2.9}$$

$$\Omega_L = \text{blockdiag}\{\Omega^{(1)}, ..., \Omega^{(L)}\} \tag{3.2.10}$$

$$\Omega_{kj}^{(i)} = y_k^{(i)} y_j^{(i)} \psi_i(x_k, x_j) + \lambda^{-1} I \tag{3.2.11}$$

The solution vectors are

$$\gamma_L = [\gamma_1; .....; \gamma_l] \tag{3.2.12}$$

$$\alpha_L = [\alpha_{1,1}; ....; \alpha_{N,1}; ....; \alpha_{1,l}; ....; \alpha_{N,l}] \tag{3.2.13}$$

This formulation is different from Oladunni & Trafalis (2005), which includes inequality constraints in its formulation, but also minimizes the L_2 norm of the error slack variable. In their study two formulations where provided, one for pairwise classification and the other for piecewise classification. For the pairwise classification the linear system is based on the following formulation (Oladunni & Trafalis, 2005):

$$\begin{aligned} &\min_{w,\gamma,\xi} \ \frac{1}{2} \sum_{i<j}^{k} \left\| w^{ij} \right\|_2^2 + \lambda \sum_{i<j}^{k} \sum_{m=1}^{l} (\xi_m^{ij})^2 \\ &s.t. \ \ \mathrm{y}^{ij} (A^{ij} w - e\gamma^{ij}) + \xi_m^{ij} \geq e, \\ &\qquad \xi_m^{ij} \geq 0 \\ &\qquad i, j = 1, \cdots, k \quad i \neq j \end{aligned} \tag{3.2.14}$$

The resulting linear system is as follows:

$$\begin{pmatrix} 0 & E^T & 0 \\ E & AA^T + \lambda^{-1} I & \lambda^{-1} I \\ 0 & \lambda^{-1} I & \lambda^{-1} I \end{pmatrix} \begin{pmatrix} \gamma \\ \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ e \\ 0 \end{pmatrix} \Rightarrow \; A_{ls} x_{ls} = b_{ls} \tag{3.2.15}$$

Here is a three class problem in matrix form:

$$A = \begin{pmatrix} A^1 & 0 & 0 \\ -A^2 & 0 & 0 \\ 0 & A^1 & 0 \\ 0 & -A^3 & 0 \\ 0 & 0 & A^2 \\ 0 & 0 & -A^3 \end{pmatrix}, \quad E = \begin{pmatrix} -e^1 & 0 & 0 \\ e^2 & 0 & 0 \\ 0 & -e^1 & 0 \\ 0 & e^3 & 0 \\ 0 & 0 & -e^2 \\ 0 & 0 & e^3 \end{pmatrix} \tag{3.2.16}$$

And solution vector

$$\gamma = \left[\gamma^{12}, \gamma^{13}, .., \gamma^{(k-1)k} \right]^T, \alpha = \left[\alpha^{12}, \alpha^{13}, .., \alpha^{(k-1)k} \right]^T \tag{3.2.17}$$

The piecewise classification linear system is based on formulation (3.2.3) and the resulting linear system is given below (Oladunni & Trafalis, 2005):

$$\begin{pmatrix} 0 & \hat{E}^T & 0 \\ \hat{E} & \frac{1}{k+1} \hat{A}\hat{A}^T + \lambda^{-1} I & \lambda^{-1} I \\ 0 & \lambda^{-1} I & \lambda^{-1} I \end{pmatrix} \begin{pmatrix} \gamma \\ \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ e \\ 0 \end{pmatrix} \Rightarrow \; \hat{A}_{ls} \hat{x}_{ls} = \hat{b}_{ls} \tag{3.2.18}$$

Here is a three class problem in matrix form:

$$\hat{A} = \begin{pmatrix} A^1 & -A^1 & 0 \\ A^1 & 0 & -A^1 \\ -A^2 & A^2 & 0 \\ 0 & A^2 & -A^2 \\ -A^3 & 0 & A^3 \\ 0 & -A^3 & A^3 \end{pmatrix}, \quad \hat{E} = \begin{pmatrix} -e^1 & e^1 & 0 \\ -e^1 & 0 & e^1 \\ e^2 & -e^2 & 0 \\ 0 & -e^2 & e^2 \\ e^3 & 0 & -e^3 \\ 0 & e^3 & -e^3 \end{pmatrix} \tag{3.2.19}$$

The solution vector

$$\gamma = \left[\gamma^1, \gamma^2, .., \gamma^k\right]^T, \alpha = \left[\alpha^{ij^T}, \alpha^{ji^T}, ..., \alpha^{(k-1)k}, \alpha^{k(k-1)}\right]^T \quad (3.2.20)$$

Problems (3.2.3) and (3.2.14) have nonnegative error constraints. However, if the error variable is negative, then the first constraint of both problems can still be satisfied if the error variable is set to zero. Setting the error variable to zero also decreases the objective function value; hence, the error constraint can be removed. In classification, the main concentration is the ability to generalize with weights obtained from a model i.e. the solution to the decision variables. The functional value of the objective function is not so much of a problem. However, if one is overly concerned about the objective function, then the nonnegative constraints can be dropped from both models (3.2.3) and (3.2.14). Hence, the last row and column of block matrices (3.2.15) and (3.2.18) won't be necessary.

Fung & Mangasarian (2005) extended the PSVM idea to the multi-classification case. This SVM formulation is called multi-category proximal support vector machine (MPSVM)—similar to the binary case—which classifies new points by assigning them to the closer of the two parallel planes that are as far apart as possible. The idea is closely aligned with the OAA method where the i^{th} class is separated from the rest.

The Linear Proximal Support Vector Machine solves the following optimization problem:

$$\min_{(w,\gamma)\in R^{n+1}} \frac{\nu}{2}\left\|D(Aw - e\gamma) - e\right\|^2 + \frac{1}{2}\left\|\begin{bmatrix} w \\ \gamma \end{bmatrix}\right\|^2 \quad (3.2.21)$$

The membership of each point in A either to a positive or negative class is specified by a $m \times m$ matrix, D, whose diagonals are labels $D_{ii} \in \{+1, -1\}$. To obtain the nonlinear classifying weights, problem (3.2.21) is modified using a dual representation of $w = A'D\alpha$, where α is a dual variable.

The Nonlinear Proximal Support Vector Machine is described through the following optimization problem:

$$\min_{(\alpha,\gamma)\in R^{m+1}} \frac{\nu}{2}\left\|D(K(A,A')D\alpha - e\gamma) - e\right\|^2 + \frac{1}{2}\left\|\begin{bmatrix}\alpha\\ \gamma\end{bmatrix}\right\|^2 \tag{3.2.22}$$

In order to obtain the classification weights, we need to solve the optimality conditions of problem (3.2.21) and (3.2.22).

3.3 Knowledge-Based Support Vector Machines

In data mining applications, incorporation of prior knowledge is usually not considered and this is due to the lack of algorithms which do not have the necessary provisions for writing explicitly, the constraints needed to represent the prior knowledge. However, if one has prior information regarding a class or label, we need to find a representation of such knowledge and incorporate it into some classification problem. By solving the modified problem we can have a more robust solution.

From 2002 – 2005, prior knowledge formulation, incorporation, and solution, have been studied in both the context of approximations (Fung & Mangasarian, 2005) and in the context of classifiers (Fung et al., 2002 & 2003; Mangasarian et al., 2004). In the approximate sense one can consider the bias, γ, of the PSVM and MPSVM, as a perturbation parameter of the normal vector w (i.e. how far it stretches the margin), thereby providing robust classifiers. For a more direct incorporation of prior knowledge to classification models, Fung et al. (2002 & 2003) developed explicit formulations for incorporating prior knowledge. The prior knowledge is in the form of multiple polyhedral sets belonging to one or more categories. The formulations are later introduced into

reformulations of the linear support vector machine and nonlinear kernel support vector machine model.

Mangasarian et al. (2004) also proposed a similar methodology for incorporating prior knowledge into a function approximation framework generated by a linear combination of linear or nonlinear kernels. For standard linear programming formulations, Mangasarian (2005) proposed a generalized class of linear programs with constraints in the form of implications (prior knowledge representation).

Chapter 4

Tikhonov Regularization Based Multi-classification Support Vector Machine

Multi-classification problems are very practical problems that need to be addressed.

The term multi-classification is referring to classifying given data into several categories. Available methods address such problems by breaking it up into smaller binary or two class problems. While this method is practical it consumes a lot of computational time primarily because of the algorithm, or method (quadratic programming), used in computing its solution.

The proposed method is intended to consider all data at once by solving a single optimization problem. This provides a simpler solution model based on linear techniques, and also provides explicit solutions to the classifying weights of the multi-class problem in terms of the given data. With the derived explicit solutions the proposed method can be implemented adequately, especially for linearly separable problems. For very large scale problems, one might have to revert back to solving several independent two class models. The decomposition of the problem can be beneficial particularly when analyzing the problem in the feature space, because the dimensionality of the problem increases tremendously there. Hence, the issues of multi-classification problems are related to the size of the problem and the computational time required in obtaining a solution. For k independently classes, several independent optimality conditions need to be satisfied. A motivating factor would be that the smaller the number of optimality conditions required

to be solved for satisfying a classification model, the lesser the computational time required in obtaining a solution to that model problem. (Please note that in chapter 4, the separating hyperplane is of the form, $f(x) = x^T w - \gamma$).

4.1 Linear Multi-classification Tikhonov Regularization Support Vector Machine: Pairwise Separation

Consider a problem of classifying data sets in R^n that are represented by a data matrix $A^{(i)} \in R^{m_i \times n}$, where $i = 1,..,k$ ($k \geq 2$ classes). Let $A^{(i)}$ be a $m_i \times n$ matrix whose rows are points in the i^{th} class, and m_i is the number of data in class *i*. Let $A^{(j)}$ be a $m_j \times n$ matrix whose rows are points in the j^{th} class, and m_j is the number of data in class *j*, and $y^{(ij)} = +1$ for class *i* and $y^{(ij)} = -1$ for class *j*. Note that we identify the classes with the matrices $A^{(i)}$ and $A^{(j)}$, and each data point can belong to exactly one class. Like in the dichotomous case, we formulate the optimal pairwise linear separator for the separable case. For the pairwise separable case there exists a $w^{(ij)} \in R^n$ and $\gamma^{(ij)} \in R$, such that

$$A^{(i)}w^{(ij)} > \gamma^{(ij)}e, \quad \gamma^{(ij)}e > A^{(j)}w^{(ij)} \quad i < j, \tag{4.1.1}$$

where *e* is a vector of ones with appropriate dimensions. Since infinitely many $w^{(ij)}$ and $\gamma^{(ij)}$ exist, the optimal solution would provide the largest margin of classification. The margin of separation between classes *i* and *j*, i.e. the distance between the halfspaces

$$A^{(i)}w^{(ij)} \geq e + \gamma^{(ij)}e \tag{4.1.2}$$

and

$$A^{(j)}w^{(ij)} \leq -e + \gamma^{(ij)}e \tag{4.1.3}$$

is $2/\|w^{(ij)}\|$. Therefore, one would minimize the reciprocal $\|w^{(ij)}\|/2$ for $i < j$.

Let $\tilde{A}^{(ij)} = \begin{bmatrix} A^{(i)} \\ A^{(j)} \end{bmatrix}$ and $y^{(ij)} = +1$ for class *i* and $y^{(ij)} = -1$ for class *j*.

For the pairwise linearly separable problem we formulate the constrained optimization problem as shown below (Trafalis & Oladunni, 2005):

$$\begin{aligned} &\min_{w,\gamma} \ \frac{1}{2}\sum_{i<j}^{k}\left\|w^{(ij)}\right\|^2 \\ &s.t. \ y^{(ij)}(\tilde{A}^{(ij)}w^{(ij)} - \gamma^{(ij)}) \geq 1, \quad i < j \end{aligned} \tag{4.1.4}$$

To classify a new point *x*, we employ the "Max Wins" strategy, this is a voting approach (Hsu & Lin, 2002; Santosa et al., 2002). For example, if the sign of [$x^T w^{(ij)} - \gamma^{(ij)}$] indicates that *x* is in the i^{th} class, then the vote for the i^{th} class is increased by one; otherwise, the j^{th} is increased by one. Hence we predict *x* as being in the class with the largest vote. In the case that those two classes have identical votes, we select the one with the smallest index.

At this juncture, we have assumed that our data set is linearly separable, but in reality that is not always the case. To accommodate inseparability, error slacks need to be included in the model and the following problem is then minimized.

$$\begin{aligned} &\min_{w,\gamma} \ \frac{\lambda_1}{2}\sum_{i<j}^{k}\left\|w^{(ij)}\right\|^2 + \frac{\lambda_2}{2}\sum_{i<j}^{k}\sum_{t=1}^{m_{(ij)}}\xi_t^{(ij)} \\ &s.t. \ y_t^{(ij)}(\tilde{A}_t^{(ij)}w^{(ij)} - \gamma^{(ij)}) + \xi_t^{(ij)} \geq 1, \quad i < j \\ &\xi_t^{(ij)} \geq 0 \quad , \quad t = 1,....,m_{(ij)} \\ &\text{where } m_{(ij)} = m_i + m_j, \ \tilde{A}^{(ij)} = \begin{pmatrix} A^{(i)} \\ A^{(j)} \end{pmatrix} \end{aligned} \tag{4.1.5}$$

Problem (4.1.5) can be considered as a two objective optimization problem, where (λ_1, λ_2) are parameters regulating the tradeoff between minimizing the L_2 norm of w (generalization ability) and the misclassification or training error, ξ. If ξ=0 then we have a linearly separable problem, else the problem is linear inseparable. The distance between the pairwise planes bounding the classes are called approximate margins (soft margins) because they provide approximate weight classifiers that allow some error.

To apply the Tikhonov regularization approach problem (4.1.4) needs to be presented in the following form:

$$\begin{aligned} &\min_{w,\gamma,\xi} \ \frac{\lambda}{2}\sum_{i<j}^{k}\left\|w^{(ij)}\right\|^2 + \frac{1}{2}\sum_{i<j}^{k}\sum_{t=1}^{m_{(ij)}}(\xi_t^{(ij)})^2 \\ &s.t. \ \ \xi_t^{(ij)} = y_t^{(ij)}(\tilde{A}_t^{(ij)}w^{(ij)} - \gamma^{(ij)}) - 1, \ \ t = 1,....,m_{(ij)}, \ \ i < j \\ &\text{where } m_{(ij)} = m_i + m_j, \ \ \tilde{A}^{(ij)} = \begin{pmatrix} A^{(i)} \\ A^{(j)} \end{pmatrix} \end{aligned} \tag{4.1.6}$$

The formulation above is similar to the RLSC (Rifkin & Klautau, 2004), with two notable differences. First, the inclusion of the bias offset constant, γ, and second, the pairwise strategy implemented in the problem above. It also bears similarity with the LS-SVM (Suykens & Vandewalle, 1999c). For this model problem an unconstrained problem is formulated.

Here is a *k* class problem rewritten in matrix notation:

Let,

$$A_{m\times n\cdot\bar{n}}=\begin{pmatrix} A^{(i)} & 0 & 0 \\ -A^{(j)} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A^{(k-1)} \\ 0 & 0 & -A^{(k)} \end{pmatrix},\ E_{m\times\bar{n}}=\begin{pmatrix} -e^{(i)} & 0 & 0 \\ e^{(j)} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & -e^{(k-1)} \\ 0 & 0 & e^{(k)} \end{pmatrix},\ e_{m\times 1}=\begin{pmatrix} e^{(i)} \\ e^{(j)} \end{pmatrix},\ \text{for } i<j \tag{4.1.7}$$

where $m=\sum_{i<j}^{k}(m_i+m_j),\ \bar{n}=k(k-1)/2$

where $A^{(i)}\in R^{m_i\times n},\ A^{(j)}\in R^{m_j\times n},\ i<j$ are matrices whose input vector belong to the i^{th} and j^{th} class; $e^{(i)}\in R^{m_i\times 1}$ and $e^{(j)}\in R^{m_j\times 1},\ i<j$ are vectors of ones. The formulation is applicable to $k\geq 2$ and can be expressed in the following matrix form:

$$\begin{aligned} &\min_{w,\gamma,\xi}\ \frac{\lambda}{2}\|w\|^2+\frac{1}{2}\|\xi\|^2 \\ &s.t.\ \ \xi=Aw+E\gamma-e \end{aligned} \tag{4.1.8}$$

λ is obtained by dividing the objective function (4.1.5) by λ_2 and setting $\lambda=\lambda_1/\lambda_2$ which accounts for the tradeoff between minimum norm and minimum error. In practice the optimal choice of λ is obtained by cross validation techniques, i.e. λ is run through a series or range of values. As λ runs through a range of values, the optimal solution of problem (4.1.8) also changes. Therefore, a specific λ corresponds to an optimal choice of *w*, while minimizing the L_2 norm of ξ. The margin between the bounding planes is still maximized with respect to *w*, however the L_2 norm of ξ is minimized instead of the L_1 norm of ξ. Note that the error slack ξ (residual) is set to the difference between the actual and estimated label, so, no nonnegative error constraint is required. The L_2 norm of ξ further provides a strong convexity of the objective function, making it easier to solve and providing reasonable estimates to the classification weights with minimum computational time.

To eliminate the error slack ξ from the constraints, problem (4.1.8) is rewritten as:

$$\min_{w,\gamma} f_{L_M T_R SVM}(w,\gamma) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{2}\|Aw + E\gamma - e\|^2 \tag{4.1.9}$$

Problem (4.1.9) is called the Linear Multi-classification Tikhonov Regularization Support Vector Machine ($L_M T_R SVM$). The modification to problem (4.1.5) is simpler to solve because explicit solutions to problem (4.1.9) in terms of the given data can be derived. The geometric representation of the $L_M T_R SVM$ model problem is illustrated in Figure 6. The planes $w^{(ij)T}x - \gamma^{(ij)} = \pm 1$ are approximate bounding planes because they allow some error (approximate margin).

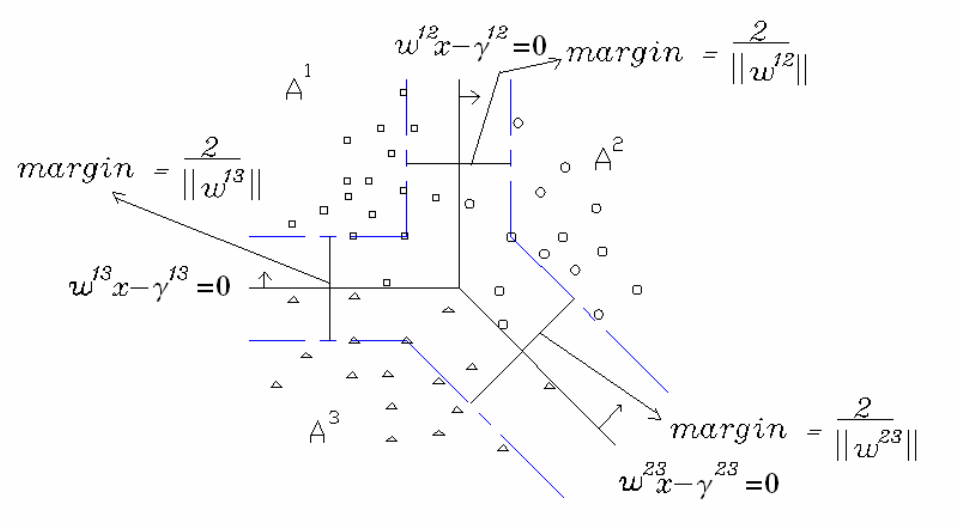

Figure 6: An illustration of the $L_M T_R SVM$ problem. The three classes, class 1, 2 and 3 are represented by small squares, circles and triangles respectively; blue lines are the approximate supporting hyperplanes for each pairwise comparison; the optimal hyperplane is orthogonal to the shortest line connecting each pairwise comparison, and intersects it halfway.

Problem (4.1.9) is a convex unconstrained optimization problem which has a unique minimum point. To find its minimum point, the optimality conditions need to be written explicitly.

Expanding problem (4.1.9)

$$f_{L_M T_R SVM}(w,\gamma) = \frac{\lambda}{2} w^T w + \frac{1}{2}[Aw + E\gamma - e]^T [Aw + E\gamma - e]$$
$$= \frac{\lambda}{2} w^T w + \frac{1}{2}[w^T A^T Aw + 2w^T A^T E\gamma - 2e^T Aw - 2\gamma^T E^T e + \gamma^T E^T E\gamma + e^T e] \quad (4.1.10)$$

Optimality conditions are obtained by setting the partial derivatives of (4.1.10) with respect to w and γ equal to zero:

$$\frac{df}{dw} = \lambda w + A^T Aw + A^T E\gamma - A^T e = 0 \quad (4.1.11)$$

$$\frac{df}{d\gamma} = E^T Aw - E^T e + E^T E\gamma = 0 \quad (4.1.12)$$

From the optimality conditions the following expressions can be derived for the optimal solution to w and γ.

From equation (4.1.12) γ (bias) can be set equal to:

$$\gamma = (E^T E)^{-1}(-E^T Aw + E^T e), \quad \text{where } \gamma = [\gamma^{(ij)}, ..., \gamma^{(k-1)k}]^T,\ i < j \quad (4.1.13)$$

Substituting (4.1.13) into (4.1.11) the following expression is obtained for w:

$$w = \left(\lambda I + A^T A - A^T E(E^T E)^{-1} E^T A\right)^{-1} \left(-A^T E(E^T E)^{-1} E^T e + A^T e\right), \quad \text{where } w = [w^{(ij)T}, ..., w^{(k-1)kT}]^T,\ i < j \quad (4.1.14)$$

Decision function is given as:

$$D(x) = sign[x^T w^{(ij)} - \gamma^{(ij)}] = \begin{cases} +1, & \text{if point } (x) \text{ is in class } A^{(i)} \\ -1, & \text{if point } (x) \text{ is in class } A^{(j)} \end{cases} \quad (4.1.15)$$

Define G and v as follows:

$$G = \lambda I + A^T A - A^T E(E^T E)^{-1} E^T A \quad (4.1.16)$$

$$\mathrm{v} = -A^T E(E^T E)^{-1} E^T e + A^T e \quad (4.1.17)$$

Assuming G is invertible, then the solution to the normal vector (4.1.14) is given as:

$$w^* = G^{-1}\mathrm{v} \tag{4.1.18}$$

and the corresponding linear system is:

$$Gw = \mathrm{v} \tag{4.1.19}$$

Solution (4.1.18) provides an estimate to the linear system of equations in (4.1.19). Such a linear system is easier to solve than the quadratic programming formulation. Methods for solving the system include matrix decomposition methods and iterative based methods. Its solution involves the inversion of a smaller dimensional matrix of the magnitude $(n*\bar{n})\times(n*\bar{n})$, where $\bar{n} = k(k-1)/2$ as opposed to QP formulation which is of the order of the number of data points and usually performs more operations to arrive at a solution. Below is an algorithm for $L_M T_R$SVM.

Algorithm 4.1 Linear Multi-classification Tikhonov Regularization SVM ($k \geq 2$)**:**

Given a data set in R^n that is represented by a matrix $A^{(i)} \in R^{m_i \times n}$, where $i = 1,..,k$ (k classes). Let $A^{(i)}$ be a $m_i \times n$ matrix whose rows are points in the i^{th} class and let $A^{(j)}$ be a $m_j \times n$ matrix whose rows are points in the j^{th} class. Weights w and γ for linear classifiers (4.1.15) are as follows.

$L_M T_R$SVM Algorithm:

Step 1: Compute G from (4.1.16) for a specific constant λ.

Step 2: Compute v from (4.1.17).

Step 3: Determine w from (4.1.18).

Step 4: Determine γ from (4.1.13).

Step 5: Classify a new point x by (4.1.15).

It is in step 3 that matrix methods or iterative methods are invoked to obtain a solution to *w*.

4.1.1 Numerical Example (AND & three class problem)

To illustrate the workings of the proposed method, two small problems are considered, the AND problem and a simple three class problem. The computations for the two small problems were conducted using MATLAB.

AND Problem, given sets:

$$A^{(1)} = \begin{pmatrix} 1 & 1 \end{pmatrix},\ A^{(2)} = \begin{pmatrix} -1 & -1 \\ -1 & 1 \\ 1 & -1 \end{pmatrix} \tag{4.1.1.1}$$

Table 1 illustrates the relationship between the inputs and outputs for the AND problem.

Table 1. Relationship between input and output of AND Problem

x_1	x_2	**AND (output)**
1	1	1
-1	-1	-1
1	-1	-1
-1	1	-1

Problem (4.1.9) is solved with the given sets in (4.1.1.1). As the tradeoff increases, the approximate margin (L_2 norm of *w*) increases (decreases). From Table 2 and Figure 7 it is evident that a specific constant corresponds to a specific margin, and it is within this approximate margin that the sum of square errors is minimized. At λ =0, the margin is smallest; at that juncture only the sum of square errors is minimized. When λ =1, both the L_2 norm of *w* and the sum of square errors are minimized. At λ =5 in bold face, the model over generalizes because the margin becomes too large. Illustrations of the approximate

classifiers are shown in Figure 7, and at λ =5 (blue dash line) there is an over forecasting classifier (solution to the norm is too small).

Because some errors are allowed, the supporting hyperplanes become approximate bounding planes for classes *i* and *j*. Also, there are no support vectors as a result of the approximation because no points lie on the supporting hyperplanes (see Figure 8). However, the approximate margin does increase as a direct consequence of the approximate bounding planes centered in the region of its categorical points, but pushed as far apart from each class (see Figure 8).

Optimal separating hyperplane for λ =0 (blue line Figure 7):

$$w^T x - \gamma = 0 \;\Rightarrow\; \begin{pmatrix} w_1 & w_2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \gamma = \begin{pmatrix} 0.5 & 0.5 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 0.5 \Rightarrow 0.5x_1 + 0.5x_2 = 0.5 \quad (4.1.1.2)$$

Table 2. $L_M T_R$SVM solution to AND Problem (varying tradeoff constant λ)

λ	w_1	w_2	γ	Margin	Empirical error
0	0.5	0.5	0.5	1.4142	0.5
0.1	0.4878	0.4878	0.5	1.4496	0.5006
0.5	0.4444	0.4444	0.5	1.591	0.5123
1	0.4	0.4	0.5	1.7678	0.54
2	0.3333	0.3333	0.5	2.1213	0.6111
3	0.2857	0.2857	0.5	2.4749	0.6837
4	0.25	0.25	0.5	2.8284	0.75
5	**0.2222**	**0.2222**	**0.5**	**3.182**	**0.8086**

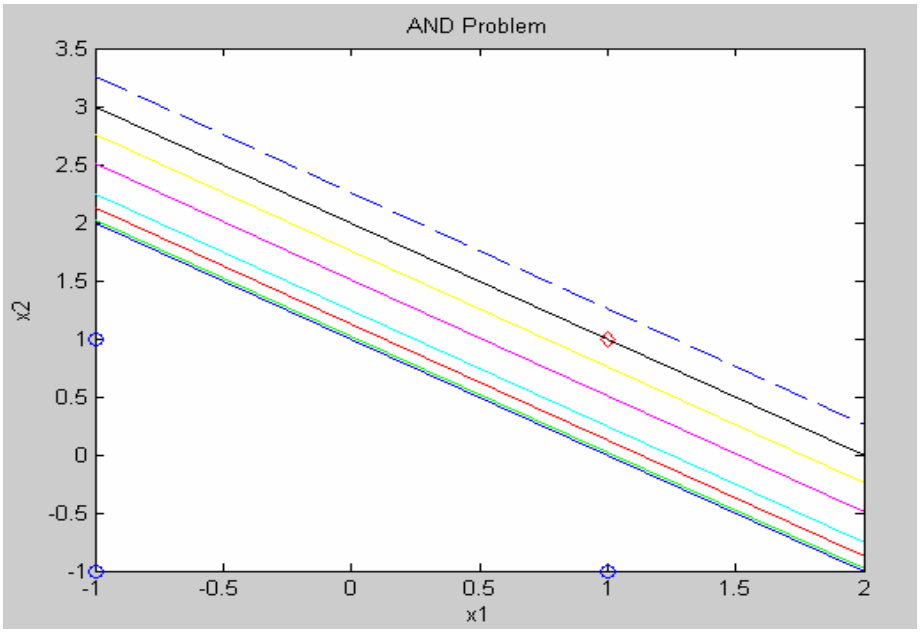

Figure 7: Illustration of L_MT_RSVM optimal hyperplanes with varying tradeoff constants for AND problem. The blue line for λ =0, green line for λ =0.1, red line for λ =0.5, cyan line for λ =1, magenta line for λ =2, yellow line for λ =3, black line for λ =4, blue dash line for λ =5. Red diamond for class A^1, blue circles for class A^2.

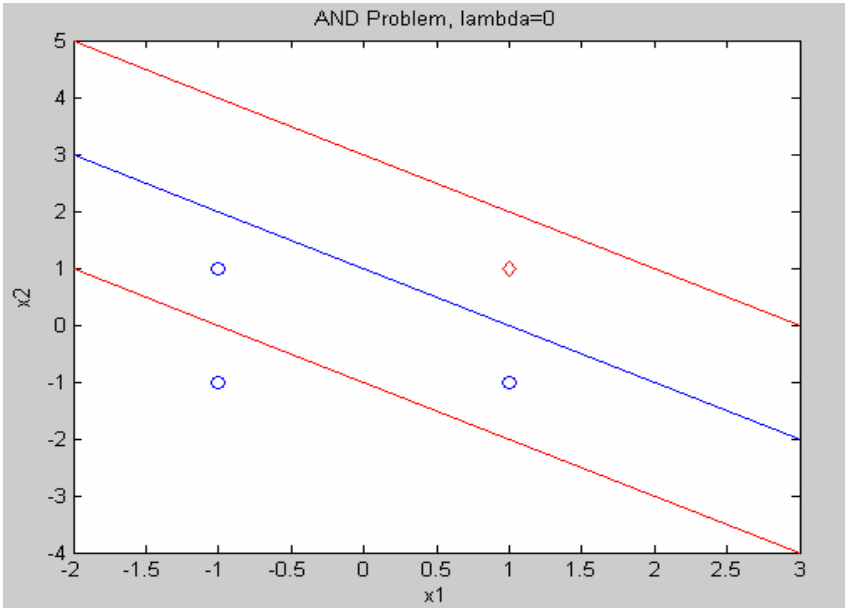

Figure 8: Illustration of L_MT_RSVM hyperplanes λ =0 for AND problem: The supporting approximate hyperplanes (with some error) are in red, the optimal hyperplane in blue. Red diamond for class A^1, blue circles for class A^2.

Given the sets below (see Figure 9):

$$A^{(1)} = \begin{pmatrix} -1 & 1 \end{pmatrix},\ A^{(2)} = \begin{pmatrix} 1 & 1 \end{pmatrix},\ A^{(3)} = \begin{pmatrix} -1 & -1 \end{pmatrix} \tag{4.1.1.3}$$

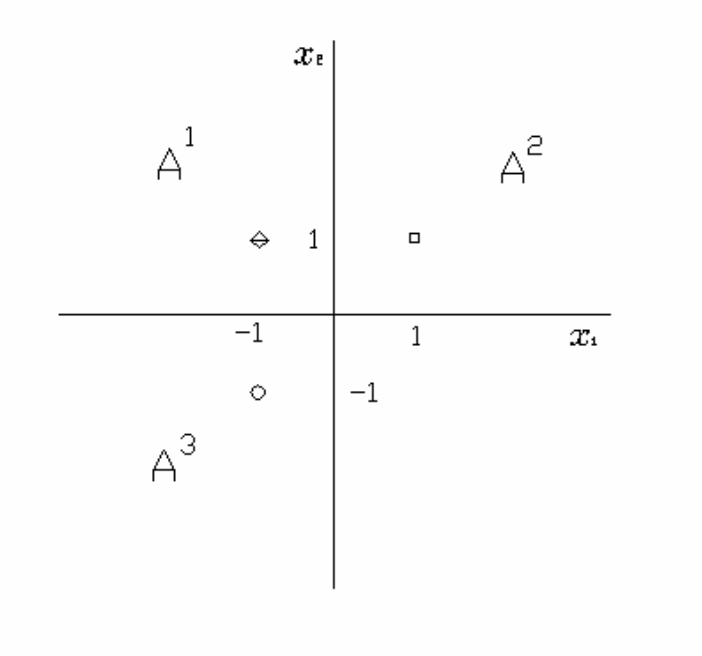

Figure 9: Illustration of THREE class problem. Diamond for class A^1, square for class A^2, circle for class A^3.

Problem (4.1.9) is solved on a simple three class problem. All the experiments for this example were performed using MATLAB.

Table 3 summarizes the classification weight statistics. Notice at λ =0, there is no solution (infeasible); this will happen because at 0, G in (4.1.16) is a singular matrix. The tradeoff constant, not only tries to find a balance between the minimum norm and error, it also forces matrix G (4.1.16) to be nonsingular (G is of full rank). Hence for this problem, the tradeoff constant should be greater than zero ($\lambda > 0$). Next we describe the optimal separating planes at λ =0.0001:

$$\text{class 1 v 2} \Rightarrow \begin{pmatrix} w_1 & w_2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \gamma = (-1 \quad 0)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 0 = -x_1 = 0$$

$$\text{class 1 v 3} \Rightarrow \begin{pmatrix} w_1 & w_2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \gamma = (0 \quad 1)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 0 = x_2 = 0 \qquad (4.1.1.4)$$

$$\text{class 2 v 3} \Rightarrow \begin{pmatrix} w_1 & w_2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \gamma = (0.5 \quad 0.5)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 0 = 0.5x_1 + 0.5x_2 = 0$$

Table 3. $L_M T_R$SVM solution to THREE class Problem (varying tradeoff constant λ)

	class 1 v 2				class 1 v 3				class 2 v 3			
λ	w_1	w_2	γ	margin	w_1	w_2	γ	margin	w_1	w_2	γ	margin
0	Infeasible											
0.0001	-1	0	0	1	0	1	0	1	0.5	0.5	0	1.4142
0.0005	-0.9998	0	0	1.0003	0	0.9998	0	1.0003	0.4999	0.4999	0	1.4144
0.001	-0.9995	0	0	1.0005	0	0.9995	0	1.0005	0.4999	0.4999	0	1.4146
0.005	-0.9975	0	0	1.0025	0	0.9975	0	1.0025	0.4994	0.4994	0	1.416
0.01	-0.995	0	0	1.005	0	0.995	0	1.005	0.4988	0.4988	0	1.4177
0.05	-0.9756	0	0	1.025	0	0.9756	0	1.025	0.4938	0.4938	0	1.4319
0.1	-0.9524	0	0	1.05	0	0.9524	0	1.05	0.4878	0.4878	0	1.4496
0.5	-0.8888	0	0	1.25	0	0.8888	0	1.25	0.4444	0.4444	0	1.591
1	-0.6667	0	0	1.5	0	0.6667	0	1.5	0.4	0.4	0	1.7678
5	-0.2857	0	0	3.5	0	0.2857	0	3.5	0.2222	0.2222	0	3.182

Similar to the AND problem, $w^{(ij)T}x - \gamma^{(ij)} = \pm 1$ as depicted in Figure 6 are approximate bounding planes separating $A^i \in R^{m_i \times n}$, where $i = 1,..,k$ ($k = 3$ classes).

The optimal separating hyperplanes are the same, however, since the pairwise comparisons are distinctive and bounded by a ± 1, the supporting hyperplanes cannot be the same. The pairwise supporting hyperplanes are approximate bounding planes. Note that only at λ =0.0001, the supporting hyperplanes touch the points (see Figures 10, 11 & 12).

As the tradeoff increases, the approximate margin (L_2 norm of w) increases (decreases). At small increments of λ, the classifiers are not significantly different and the margins are much closer. Values of λ between 0.0005 and 0.01 give similar approximate hyperplanes that minimize the errors. At larger increments of λ, the influence on the classifiers becomes obvious. While the optimal weights gives good generalization, its supporting

hyperplanes move further away from the optimal hyperplane and data points belonging to its respective class (see Figures 10, 11 & 12).

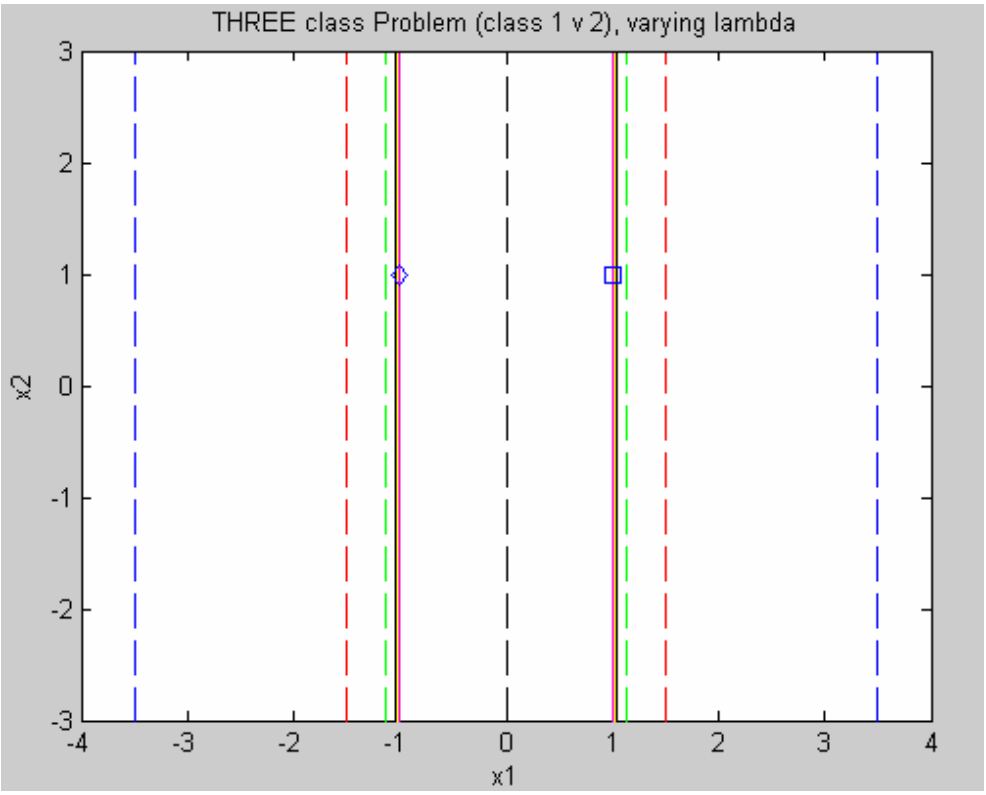

Figure 10: Illustration of $L_M T_R$SVM hyperplane with varying tradeoff constants for THREE class problem. The optimal hyperplane in black dashed line, supporting hyperplanes red line (λ =0.0001), blue line (λ =0.0005), green line (λ =0.001), cyan line (λ =0.005), magenta line (λ =0.01), yellow line (λ =0.05), black line (λ =0.1), green dash line (λ =0.5), red dash line (λ =1), blue dash line (λ =5). Diamond for class A^1, square for class A^2.

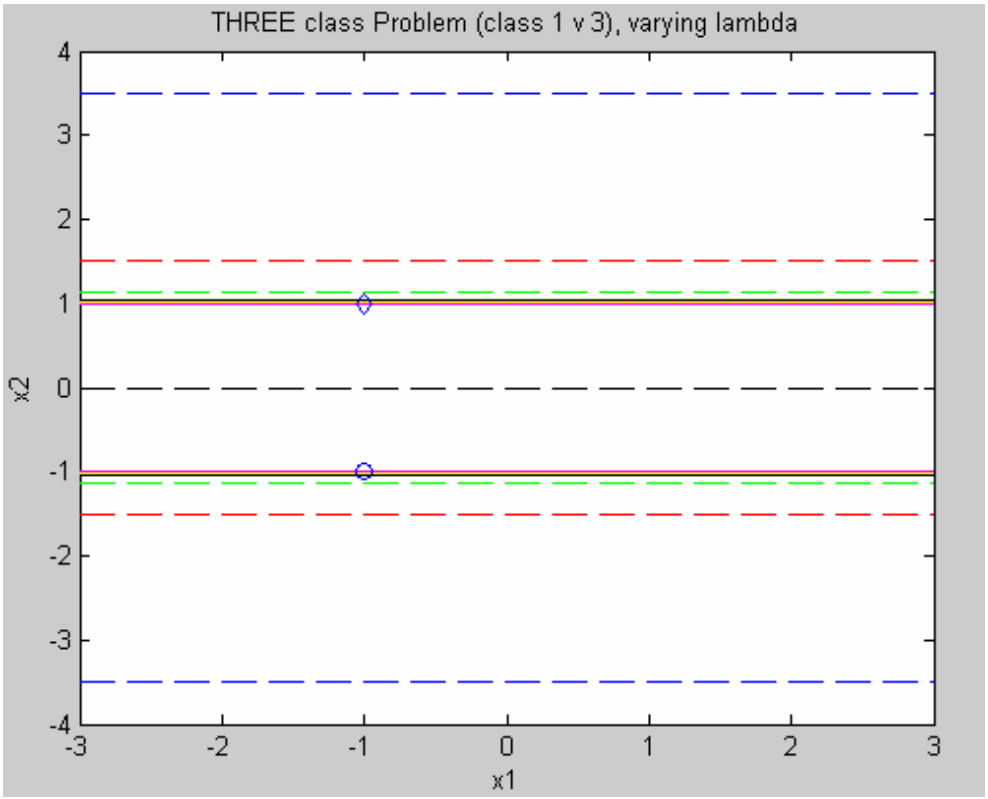

Figure 11: Illustration of $L_M T_R$SVM hyperplane with varying tradeoff constants for THREE class problem. The optimal hyperplane in black dashed line, supporting hyperplanes red line (λ =0.0001), blue line (λ =0.0005), green line (λ =0.001), cyan line (λ =0.005), magenta line (λ =0.01), yellow line (λ =0.05), black line (λ =0.1), green dash line (λ =0.5), red dash line (λ =1), blue dash line (λ =5). Diamond for class A^1, circle for class A^3.

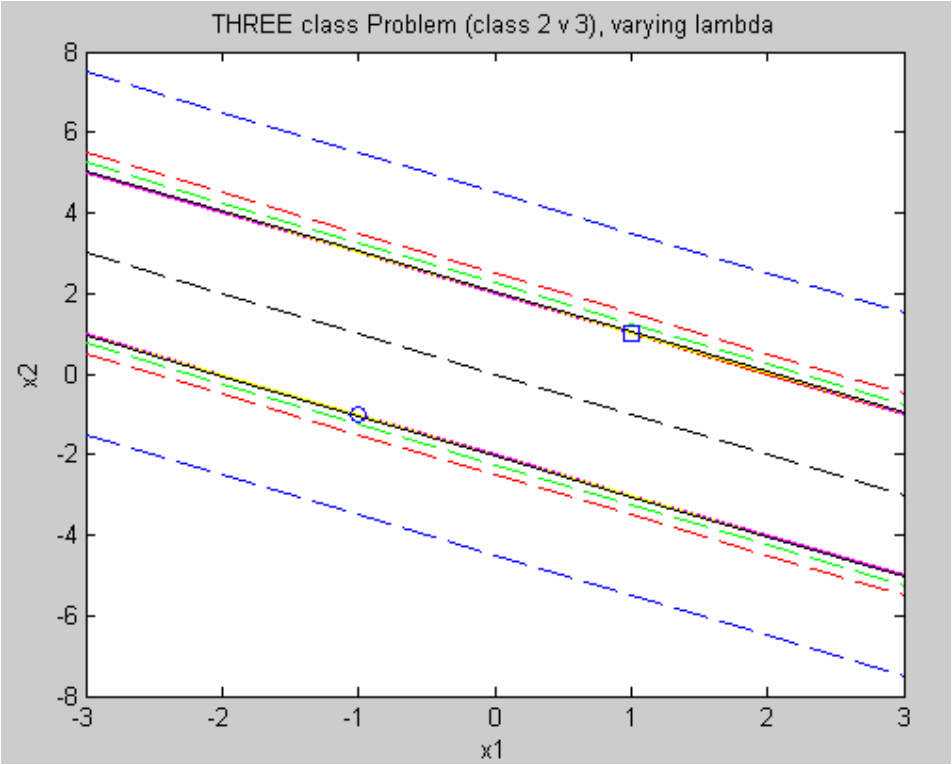

Figure 12: Illustration of $L_M T_R$SVM hyperplane with varying tradeoff constants for THREE class problem. The optimal hyperplane in black dashed line, supporting hyperplanes red line (λ =0.0001), blue line (λ =0.0005 & 0.001), cyan line (λ =0.005), magenta line (λ =0.01), yellow line (λ =0.05), black line (λ =0.1), green dash line (λ =0.5), red dash line (λ =1), blue dash line (λ =5). Square for class A^2, circle for class A^3.

In Figures 10, 11 & 12, the black dash line is the optimal hyperplane—the line perpendicular to the normal vector w—for separation between classes 1 & 2, classes 1 & 3, and classes 2 & 3 respectively. Since the optimal hyperplanes are the same irrespective of the tradeoff constant for a specific pairwise comparison, only the optimal classifier with λ =0.0001 is depicted on Figures 10, 11 & 12 together with the bounding planes for the varying tradeoff constants. As the tradeoff increases the approximate bounding planes move far apart from the optimal hyperplane and points unique to the bounding planes. For large values of λ the margin becomes too wide thereby increasing the error margin of the point from its position to its respective approximate bounding plane.

4.2 Nonlinear Multi-classification Tikhonov Regularization Kernel Machine: Pairwise Separation

To obtain nonlinear classifiers it is essential to carry out a nonlinear mapping from the input space to a feature space (Burges, 1998; Cristianini & Shawe-Taylor, 2000) using a mapping function (see Fig. 3),

$$\phi : R^n \to F \tag{4.2.1}$$

But since we do not know the mapping function, a kernel function is employed to implicitly compute the inner products of the input vectors in feature space. Depending on the specific kernel function chosen, the resulting kernel matrix defines the similarity or dissimilarity of the input vectors. The kernel matrix can also be considered a distance matrix. To formulate the nonlinear counterpart of the linear classification, the primal variable *w* is replaced by its equivalent dual representation as shown below:

$$w = \overline{A}^T Y \alpha \tag{4.2.2}$$

Then problem (4.1.9) can be modified as follows:

$$\begin{aligned} \min_{\alpha,\gamma} f_{L_M T_R KM}(\alpha,\gamma) &= \frac{\lambda}{2}\left\|\overline{A}^T Y\alpha\right\|^2 + \frac{1}{2}\left\|Y(\overline{A}\overline{A}^T Y\alpha - \overline{E}\gamma) - e\right\|^2 \\ &= \frac{\lambda}{2}\alpha^T Y\overline{A}\overline{A}^T Y\alpha + \frac{1}{2}\left\|Y(\overline{A}\overline{A}^T Y\alpha - \overline{E}\gamma) - e\right\|^2 \end{aligned} \tag{4.2.3}$$

Here is a *k* class problem in matrix form:

$$\overline{A}_{m\times n\cdot\overline{n}} = \begin{pmatrix} A^{(i)} & 0 & 0 \\ A^{(j)} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A^{(k-1)} \\ 0 & 0 & A^{(k)} \end{pmatrix}, \ \overline{E}_{m\times\overline{n}} = \begin{pmatrix} e^{(i)} & 0 & 0 \\ e^{(j)} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & e^{(k-1)} \\ 0 & 0 & e^{(k)} \end{pmatrix}, Y_{m\times m} = \begin{pmatrix} Y^{(12)} & 0 & 0 \\ 0 & Y^{(ij)} & 0 \\ 0 & 0 & Y^{(k-1)k} \end{pmatrix}, \tag{4.2.4}$$

$$e_{m\times 1} = \begin{pmatrix} e^{(i)} \\ e^{(j)} \end{pmatrix} \text{ for } i < j, \text{ where } m = \sum_{i<j}^{k}(m_i + m_j), \ \overline{n} = k(k-1)/2$$

where $A^{(i)} \in R^{m_i \times n}$, $A^{(j)} \in R^{m_j \times n}$, $i < j$ are matrices whose input vector belong to the i^{th} and j^{th} class; $e^{(i)} \in R^{m_i \times 1}$ and $e^{(j)} \in R^{m_j \times 1}$, $i < j$ are vectors of ones; Y^{ij} are diagonal matrices whose diagonals are ± 1 for classes *i* and *j* respectively, and α is the dual variable.

Problem (4.2.3) is still a linear classification problem, because $\overline{A}\overline{A}^T$ is a linear kernel matrix. But, replacing the linear kernel by a nonlinear kernel $K(\overline{A}, \overline{A}^T)$, where $K(\overline{A}, \overline{A}^T)$ is a $m \times m$ matrix, problem (4.2.3) then becomes:

$$\min_{\alpha,\gamma} f_{NL_M T_R KM}(\alpha, \gamma) = \frac{\lambda}{2} \alpha^T Y K(\overline{A}, \overline{A}^T) Y \alpha + \frac{1}{2} \left\| Y(K(\overline{A}, \overline{A}^T) Y \alpha - \overline{E}\gamma) - e \right\|^2 \quad (4.2.5)$$

Setting

$$K_{m \times m} = K(\overline{A}, \overline{A}^T) \text{ and } \overline{K}_{m \times m} = YKY \quad (4.2.6)$$

where $\overline{K}$ is a square symmetric matrix.

Problem (4.2.5) is given as:

$$\min_{\alpha,\gamma} f_{NL_M T_R KM}(\alpha, \gamma) = \frac{\lambda}{2} \alpha^T \overline{K} \alpha + \frac{1}{2} \left\| \overline{K}\alpha - Y\overline{E}\gamma - e \right\|^2 \quad (4.2.7)$$

Definition 4.2 *Let* $A \in R^{m \times n}$ *and* $B \in R^{n \times k}$. *The* ***kernel*** $K(A, B)$ *maps* $R^{m \times n} \times R^{n \times k}$ *into* $R^{m \times k}$.

Definition 4.2 strongly relies on Mercer's condition (Burges, 1998; Cristianini & Shawe-Taylor, 2000) for symmetric kernel matrices.

Problem (4.2.5) is called the Nonlinear Multi-classification Tikhonov Regularization Kernel Machine (NL_MT_RKM). It is a modification of problem (4.1.9) to accommodate nonlinearly separable patterns, and explicit solutions can be derived in dual space in terms of the given data.

The NL_MT_RKM model is a convex unconstrained optimization problem which has a unique minimum point. To find its minimum point, the optimality conditions need to be written explicitly.

Expanding the second term in problem (4.2.7)

$$\begin{aligned}\frac{1}{2}\left\|\bar{K}\alpha - Y\bar{E}\gamma - e\right\|^2 &= \frac{1}{2}\left[\bar{K}\alpha - Y\bar{E}\gamma - e\right]^T\left[\bar{K}\alpha - Y\bar{E}\gamma - e\right] \\ &= \frac{1}{2}\left[\alpha^T\bar{K}\bar{K}\alpha - 2\alpha^T\bar{K}Y\bar{E}\gamma - 2e^T\bar{K}\alpha + 2\gamma^T\bar{E}^TYe + \gamma^T\bar{E}^TYY\bar{E}^T\gamma + e^Te\right]\end{aligned} \tag{4.2.8}$$

Optimality conditions are obtained by setting the partial derivatives of (4.2.7) equal to zero, with respect to α and γ:

$$\frac{df}{d\alpha} = \lambda\bar{K}\alpha + \bar{K}\bar{K}\alpha - \bar{K}Y\bar{E}\gamma - \bar{K}^Te = 0 \tag{4.2.9}$$

$$\frac{df}{d\gamma} = -\bar{E}^TY\bar{K}\alpha + \bar{E}^TYe + \bar{E}^TYY\bar{E}\gamma = 0 \tag{4.2.10}$$

From the optimality conditions, explicit solutions to the classification parameters, α and γ, can be derived.

From equation (4.2.10) γ (bias) can be set equal to:

$$\begin{aligned}&\gamma = (\bar{E}^TYY\bar{E})^{-1}(\bar{E}^TY\bar{K}\alpha - \bar{E}^TYe), \\ &\text{where } \gamma = [\gamma^{(ij)},....,\gamma^{(k-1)k}]^T,\ i<j\end{aligned} \tag{4.2.11}$$

Substituting (4.2.11) in (4.2.9) the following expression is obtained for α:

$$\begin{aligned}&\alpha = \left(\lambda I + \bar{K} - Y\bar{E}\left(\bar{E}^TYY\bar{E}\right)^{-1}\bar{E}^TY\bar{K}\right)^{-1}\left(-Y\bar{E}\left(\bar{E}^TYY\bar{E}\right)^{-1}\bar{E}^TYe + e\right), \\ &\text{where } \alpha = [\alpha^{(ij)T},....,\alpha^{(k-1)kT}]^T,\ i<j\end{aligned} \tag{4.2.12}$$

Decision function is given as:

$$D(x) = sign\left[K(x^T, A^{(ij)T})Y^{(ij)}\alpha^{(ij)} - \gamma^{(ij)} \right] = \begin{cases} +1, & \text{if point } (x) \text{ is in class } A^{(i)} \\ -1, & \text{if point } (x) \text{ is in class } A^{(j)} \end{cases} \quad (4.2.13)$$

To classify a new point x, the voting approach strategy (Santosa, et al., 2002; Hsu & Lin, 2002) is employed. For example, if sign $[K(x^T, A^{(ij)T})Y^{(ij)}\alpha^{(ij)} - \gamma^{(ij)}]$ says that x is in the i^{th} class, then the vote for the i^{th} class is increased by one. Otherwise, the j^{th} is increased by one. Then we predict x as being in the class with the largest vote. In the case that those two classes have identical votes, select the one with the smallest index.

Define $\bar{G}$ and $\bar{v}$ as follows:

$$\bar{G} = \lambda I + \bar{K} - Y\bar{E}\left(\bar{E}^T YY\bar{E}\right)^{-1} \bar{E}^T Y\bar{K} \quad (4.2.14)$$

$$\bar{\mathrm{v}} = -Y\bar{E}\left(\bar{E}^T YY\bar{E}\right)^{-1} \bar{E}^T Ye + e \quad (4.2.15)$$

Assuming G_α is invertible then the solution to the dual vector (4.2.12) is given as:

$$\alpha^* = \bar{G}^{-1}\bar{\mathrm{v}} \quad (4.2.16)$$

The corresponding linear system is:

$$\bar{G}\alpha = \bar{\mathrm{v}} \quad (4.2.17)$$

Solution (4.2.16) provides a solution to the linear system of equations in (4.2.17). Such a linear system is easier to solve than the quadratic programming formulation. Methods for solving the system include matrix decomposition methods or iterative based methods. Its solution involves the inversion of an $m \times m$ matrix, where $m = \sum_{i<j}^{k} (m_i + m_j)$. Below is an algorithm for NL_MT_RKM.

Algorithm 4.2 Nonlinear Multi-classification Tikhonov Regularization Kernel Machine ($k \geq 2$):

Given a data set in R^n that is represented by a matrix $A^{(i)} \in R^{m_i \times n}$, where $i = 1,..,k$ (k classes). Let $A^{(i)}$ be a $m_i \times n$ matrix whose rows are points in the i^{th} class and let $A^{(j)}$ be a $m_j \times n$ matrix whose rows are points in the j^{th} class. Weights α and γ for the nonlinear classifier (4.2.13) is as follows:

NL_MT_RKM Algorithm:

Step 1: Choose an appropriate kernel function such as a polynomial kernel function (2.1.3.4) or a Gaussian rbf kernel (2.1.3.5), then compute its square symmetric kernel matrix.

Step 2: Compute $\bar{G}$ from (4.2.14) for a specific constant λ.

Step 3: Compute $\bar{v}$ from (4.2.15).

Step 4: Determine α from (4.2.16).

Step 5: Determine γ from (4.2.11).

Step 6: Classify a new point *x* by (4.2.13).

It is in step 4 that matrix methods or iterative methods are invoked to obtain a solution to α.

4.2.1 Numerical Example – XOR problem

Given the sets:

Table 4 illustrates the relationship between the inputs and outputs for the XOR problem. The XOR problem was trained with a polynomial kernel with degree 2 and computations were conducted using MATLAB.

Table 4. Relationship between input and output of XOR Problem

x_1	x_2	XOR (output)
1	1	1
-1	-1	1
1	-1	-1
-1	1	-1

In the XOR Problem, the data matrices are as follows:

$$A^{(1)} = \begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}, \; A^{(2)} = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} \tag{4.2.1.1}$$

Problem (4.2.5) is solved with the given sets in (4.2.1.1). From Table 5, it is evident that an increase in the tradeoff constant decreases the solution value of the dual variable. At λ =0, the problem is infeasible, indicating that $\bar{G}$ matrix is not invertible. Therefore, only for a positive constant there exists a solution to (4.2.16) for the given sets (4.2.1.1). When λ =0.001, we obtain the same solution to the traditional SVM; with an increase of the constant λ, the dual vector solution decreases. At λ =100 and greater, the model can over generalize because the solution to the dual variables gets smaller, implying that too much emphasis has been placed on the generalization ability and little or no attention is placed on reducing the training or misclassification error.

Decision separating hyperplane for λ =0.001 (see Fig. 13):

$$\begin{aligned} D(x) &= sign\left[K(x^T, A^{(ij)T}) Y^{(ij)} \alpha^{(ij)} - \gamma^{(ij)} \right] \\ D(x) &= sign\left[\left(x^T A^{(12)T} + 1 \right)^2 Y^{(12)} \alpha^{(12)} - \gamma^{(12)} \right] \\ D(x) &= sign\left[x_1 x_2 \right] \end{aligned} \tag{4.2.1.2}$$

Table 5. NL_MT_RKM solution to XOR Problem (varying tradeoff constant λ)

λ	α_1	α_2	α_3	α_4	γ
0	Infeasible				
0.001	0.125	0.125	0.125	0.125	0
0.01	0.1248	0.1248	0.1248	0.1248	0
0.1	0.1235	0.1235	0.1235	0.1235	0
1	0.1111	0.1111	0.1111	0.1111	0
5	0.0909	0.0909	0.0909	0.0909	0
10	0.0556	0.0556	0.0556	0.0556	0
100	0.0093	0.0093	0.0093	0.0093	0

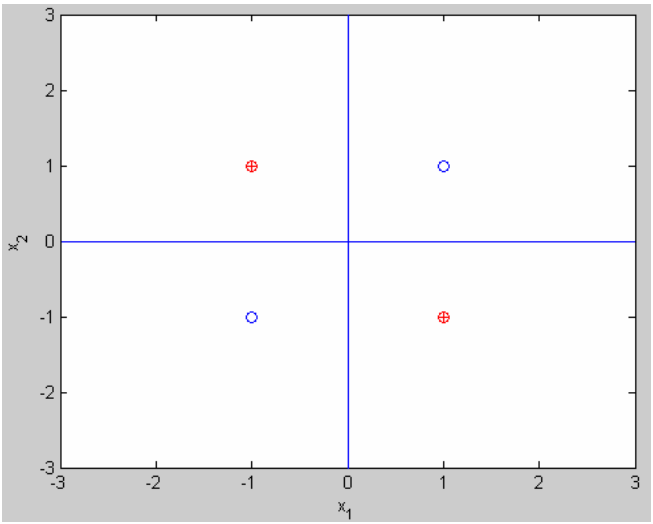

Figure 13: Illustration of NL_MT_RKM hyperplanes λ =0.001 for XOR discrimination problem with decision function $D(x) = sign[x_1 x_2]$ (4.2.1.2). Blue circles for class A^1, red circles for class A^2.

4.3 Reduced Nonlinear Kernel Multi-classification Tikhonov Regularization Machine: Pairwise Separation

The optimization problem of the NL_MT_RKM model (4.2.7) requires the entire training data set which results in a kernel matrix $K(\bar{A}, \bar{A}^T)$ with *m* rows and *m* dual variables. If the number of the data in the training set increases, so will its kernel matrix, and this will

impose a heavy burden on the computational time used in obtaining a solution. Similar problems will appear in memory storage of parameters and solution, and/or even numerical instability. In order to reduce the computing time used to generate nonlinear classifiers and the amount of memory usage, an algorithm that uses a small subset of the original data set is proposed. To generate nonlinear classifiers, the whole training dataset is used as the constraints, and a small subset of the training set is used as the dual variables of the optimization problem. This can be achieved by applying definition 4.2 without making any assumption to create a rectangular $m \times \hat{m}$ kernel $K(\overline{A}, \hat{A}^T)$. A similar approach was developed by Lee & Mangasarian (2001).

Such a kernel matrix reduces the size of the optimization problem to be solved and simplifies the description of the pairwise nonlinear separating surfaces.

Since this reduces the number of dual variables, $\overline{A}^T$ can be replaced by $\hat{A}^T$ in the dual representation of *w* as shown below:

$$w = \hat{A}^T \hat{Y} \hat{\alpha}$$

$$\text{where } \hat{A}_{\hat{m} \times n \cdot \overline{n}},\ \hat{Y}_{\hat{m} \times \hat{m}},\ \hat{m} = \sum_{i<j}^{k} (\hat{m}_i + \hat{m}_j),\ \overline{n} = k(k-1)/2 \qquad (4.3.1)$$

where $\hat{A}$ is a subset matrix similar to $\overline{A}$ in (4.2.4), but based on a percentage of the training set, i.e. $\hat{A}$ is a subset matrix containing $\hat{A}^{(i)} \in R^{\hat{m}_i \times n}$, $\hat{A}^{(j)} \in R^{\hat{m}_j \times n}$, $i < j$ whose row vectors belong to the i^{th} and j^{th} subset; $\hat{Y}$ is a diagonal matrix similar to Y in (4.2.4), but whose Y^{ij} matrix diagonals are ± 1 for classes *i* and *j* subsets respectively, and α is the dual variable.

Consider minimizing the L_2 norm of $\hat{\alpha}$, and replacing $\bar{A}^T$ by $\hat{A}^T$ then problem (4.2.5) can be expressed as:

$$\min_{\hat{\alpha},\gamma} f_{RK_M T_R M}(\hat{\alpha},\gamma) = \frac{\lambda}{2}\hat{\alpha}^T\hat{\alpha} + \frac{1}{2}\left\| Y(K(\bar{A},\hat{A}^T)\hat{Y}\hat{\alpha} - \bar{E}\gamma) - e \right\|^2 \qquad (4.3.2)$$

Setting

$$K_{m\times\hat{m}} = K(\bar{A},\hat{A}^T) \text{ and } \hat{K}_{m\times\hat{m}} = YK\hat{Y} \qquad (4.3.3)$$

where $\hat{K}$ is a rectangular matrix.

Problem (4.3.2) is given as:

$$\min_{\hat{\alpha},\gamma} f_{RK_M T_R M}(\hat{\alpha},\gamma) = \frac{\lambda}{2}\hat{\alpha}^T\hat{\alpha} + \frac{1}{2}\left\| \hat{K}\hat{\alpha} - Y\bar{E}\gamma - e \right\|^2 \qquad (4.3.4)$$

Problem (4.3.4) is called the Reduced Nonlinear Kernel Multi-classification Tikhonov Regularization Machine (RK_MT_RM). It is a variant of problem (4.2.7) to accommodate, or provide a sparse solution to the optimization problem. Explicit solutions in dual space in terms of the given data can be derived.

An issue with the least square SVMs is that sparsity is lost, due to the nonzero components of the dual variable. In obtaining a sparse solution we aim to find a solution with the least number of points that characterize the normal vector and bias parameter. The formulation of traditional SVMs is to find a sparse solution because its error slacks are nonnegative. However, its least square counter part is not capable of finding a sparse solution, because its error slacks are unrestricted. The proposed RK_MT_RM model can address such a problem of sparsity because few points are used to perform the mapping from the primal space to the dual space (i.e., the points used to perform the mapping

become the number of variables). Since sparsity is governed by the number of features used in training a model problem, a decrease in that number should provide a more sparse solution to *w*.

The RK_MT_RM model is a convex unconstrained optimization problem which has a unique minimum point. To find its minimum point, the optimality conditions need to be written explicitly.

Expanding the second term in problem (4.3.4)

$$\begin{aligned}\frac{1}{2}\left\|\hat{K}\hat{\alpha}-Y\bar{E}\gamma-e\right\|^2 &= \frac{1}{2}\left[\hat{K}\hat{\alpha}-Y\bar{E}\gamma-e\right]^T\left[\hat{K}\hat{\alpha}-Y\bar{E}\gamma-e\right] \\ &= \frac{1}{2}\left[\hat{\alpha}^T\hat{K}^T\hat{K}\hat{\alpha}-2\hat{\alpha}^T\hat{K}^TY\bar{E}\gamma-2e^T\hat{K}\hat{\alpha}+2\gamma^T\bar{E}^TYe+\gamma^T\bar{E}^TY^TY\bar{E}^T\gamma+e^Te\right]\end{aligned} \tag{4.3.5}$$

Optimality conditions are obtained by setting the partial derivatives of (4.3.5) equal to zero, with respect to $\hat{\alpha}$ and γ:

$$\frac{df}{d\hat{\alpha}} = \lambda\hat{\alpha}+\hat{K}^T\hat{K}\hat{\alpha}-\hat{K}^TY\bar{E}\gamma-\hat{K}^Te=0 \tag{4.3.6}$$

$$\frac{df}{d\gamma} = -\bar{E}^TY^T\hat{K}\hat{\alpha}+\bar{E}^TY^Te+\bar{E}^TY^TY\bar{E}\gamma=0 \tag{4.3.7}$$

From the optimality conditions, explicit solutions to the classification parameters, $\hat{\alpha}$ and γ, can be derived.

From equation (4.3.7) γ (bias) can be set equal to:

$$\begin{aligned}&\gamma=(\bar{E}^TY^TY\bar{E})^{-1}(\bar{E}^TY^T\hat{K}\hat{\alpha}-\bar{E}^TY^Te), \\ &\text{where } \gamma=[\gamma^{(ij)},....,\gamma^{(k-1)k}]^T,\ i<j\end{aligned} \tag{4.3.8}$$

Substituting (4.3.8) into (4.3.6) the following expression is obtained for α:

$$\begin{aligned}&\hat{\alpha}=\left(\lambda I+\hat{K}^T\hat{K}-\hat{K}^TY\bar{E}\left(\bar{E}^TY^TY\bar{E}\right)^{-1}\bar{E}^TY^T\hat{K}\right)^{-1}\left(-\hat{K}^TY\bar{E}\left(\bar{E}^TY^TY\bar{E}\right)^{-1}\bar{E}^TY^Te+\hat{K}^Te\right), \\ &\text{where } \hat{\alpha}=[\hat{\alpha}^{(ij)T},....,\hat{\alpha}^{(k-1)kT}]^T,\ i<j\end{aligned} \tag{4.3.9}$$

Decision function is given as:

$$D(x) = sign\left[K(x^T, \hat{A}^{(ij)T})\hat{Y}^{(ij)}\hat{\alpha}^{(ij)} - \gamma^{(ij)} \right] = \begin{cases} +1, & \text{if point } (x) \text{ is in class } A^{(i)} \\ -1, & \text{if point } (x) \text{ is in class } A^{(j)} \end{cases} \quad (4.3.10)$$

To classify a new point x, the voting approach strategy (Santosa, et al., 2002; Hsu & Lin, 2002) is employed. For example, if sign [$K(x^T, \hat{A}^{(ij)T})\hat{Y}^{(ij)}\hat{\alpha}^{(ij)} - \gamma^{(ij)}$] says that x is in the i^{th} class, then the vote for the i^{th} class is increased by one. Otherwise, the j^{th} is increased by one. Then we predict x as being in the class with the largest vote. In the case that those two classes have identical votes, select the one with the smallest index.

Define $\hat{G}$ and $\hat{v}$ as follows:

$$\hat{G} = \lambda I + \hat{K}^T\hat{K} - \hat{K}^T Y\bar{E}\left(\bar{E}^T Y^T Y\bar{E}\right)^{-1}\bar{E}^T Y^T\hat{K} \quad (4.3.11)$$

$$\hat{\mathrm{v}} = -\hat{K}^T Y\bar{E}\left(\bar{E}^T Y^T Y\bar{E}\right)^{-1}\bar{E}^T Y^T e + \hat{K}^T e \quad (4.3.12)$$

Assuming $\hat{G}$ is invertible; the solution to dual vector (4.3.9) is given as:

$$\hat{\alpha}^* = \hat{G}^{-1}\hat{\mathrm{v}}, \quad (4.3.13)$$

and the corresponding linear system is:

$$\hat{G}\hat{\alpha} = \hat{\mathrm{v}} \quad (4.3.14)$$

Solution (4.3.13) provides an estimate to the linear system of equations in (4.3.14). Such a linear system is easier to solve than the quadratic programming formulation. Methods for solving the system include matrix decomposition methods and iterative based methods. Its solution involves the inversion of a reduced dimensional matrix of magnitude $\hat{m} \times \hat{m}$, where $\hat{m} = \sum_{i<j}^{k} (\hat{m}_i + \hat{m}_j)$. Below is an algorithm for RK_MT_RM.

Algorithm 4.3 Reduced Nonlinear Kernel Multi-classification Tikhonov Regularization Machine ($k \geq 2$):

Given a data set in R^n that are represented by a matrix $A^{(i)} \in R^{m_i \times n}$, where $i = 1,..,k$ (k classes). Let $A^{(i)}$ be a $m_i \times n$ matrix whose rows are points in the i^{th} class and let $A^{(j)}$ be a $m_j \times n$ matrix whose rows are points in the j^{th} class. Weights $\hat{\alpha}$ and γ for nonlinear classifier (4.3.10) is as follows.

RK_MT_RM Algorithm:

Step 1: Choose a random subset matrix $\hat{A}^{(i)} \in R^{\hat{m}_i \times n}$, $\hat{A}^{(j)} \in R^{\hat{m}_j \times n}$, $i < j$ of the original data matrix $A^{(i)} \in R^{m_i \times n}$, $A^{(j)} \in R^{m_j \times n}$, $i < j$.

Step 2: Choose an appropriate kernel function such as a polynomial kernel function (2.1.3.4) or a Gaussian rbf kernel (2.1.3.5). Then compute its rectangular kernel matrix.

Step 3: Compute $\hat{G}$ from (4.3.11) for a specific constant λ .

Step 4: Compute $\hat{v}$ from (4.3.12).

Step 5: Determine $\hat{\alpha}$ from (4.3.13).

Step 6: Determine γ from (4.3.8).

Step 7: Classify a new point x by (4.3.10).

Chapter 5

Tikhonov Regularization Knowledge-Based Multi-classification Support Vector Machine

Robust solutions are solutions that investigate two types of problem; one addresses problems of uncertainty (unknown information) i.e. presence of noise in a given data set; while the other addresses problems which have prior knowledge (known information) i.e. incorporation of additional constraints to a given set. Below is the description of a model problem called Tikhonov Regularization Knowledge-Based Support Vector Machine for multi-category discrimination (Please note that in chapter 5, the separating hyperplane is of the form, $f(x) = x^T w - \gamma$).

5.1 Linear Multi-classification Tikhonov Regularization Knowledge-based Support Vector Machine: Pairwise knowledge set formulation

Consider a problem of classifying data sets in R^n that are represented by a data matrix $A^{(i)} \in R^{m_i \times n}$, where $i = 1,..,k$ ($k \geq 2$ classes). Let $A^{(i)}$ be a $m_i \times n$ matrix whose rows are points in the i^{th} class, and m_i is the number of data in class *i*. Let $A^{(j)}$ be a $m_j \times n$ matrix whose rows are points in the j^{th} class, and m_j is the number of data in class *j*, and $y^{(ij)} = +1$ for class *i* and $y^{(ij)} = -1$ for class *j*. Note that we identify the two classes with the matrices $A^{(i)}$ and $A^{(j)}$, and each data point can belong to exactly one class. The classification optimization problem is given below:

$$\begin{aligned}
&\min_{w,\gamma,\xi} \ \frac{\lambda}{2}\sum_{i<j}^{k}\left\|w^{(ij)}\right\|^2 + \frac{1}{2}\sum_{i<j}^{k}\sum_{t=1}^{m_{(ij)}}(\xi_t^{(ij)})^2 \\
&s.t. \ \ \xi_t^{(ij)} = y_t^{(ij)}(\tilde{A}_t^{(ij)}w^{(ij)} - \gamma^{(ij)}) - 1, \ \ t = 1,....,m_{(ij)}, \ \ i<j \\
&\text{where } m_{(ij)} = m_i + m_j, \ \ \tilde{A}^{(ij)} = \begin{pmatrix} A^{(i)} \\ A^{(j)} \end{pmatrix}
\end{aligned} \quad , \tag{5.1.1}$$

λ is the tradeoff constant, and $\xi^{(ij)}$ are error slack measuring the deviation of points from their respective bounding planes. The classification weights ($w^{(ij)}$, $\gamma^{(ij)}$) that characterize the optimal separating plane are shown in Figure 8.

$$x^T w^{(ij)} = \gamma^{(ij)}, \quad i<j \tag{5.1.2}$$

As shown in Figure 4, $w^{(ij)}$ is the normal vector perpendicular to the approximate bounding planes:

$$x^T w^{(ij)} = 1 + \gamma^{(ij)}, \quad x^T w^{(ij)} = -1 + \gamma^{(ij)}, \quad i<j \tag{5.1.3}$$

that bound most of the points belonging to sets $A^{(i)} \in R^{m_i \times n} \ \ i = 1,..,k$ as follows:

$$A^{(i)}w^{(ij)} \geq e + \gamma^{(ij)}e, \quad A^{(j)}w^{(ij)} \leq -e + \gamma^{(ij)}e, \quad i<j \tag{5.1.4}$$

where e is a vector of ones with appropriate dimension. The locations of the approximate bounding planes relative to the origin are determined by the value of $\gamma^{(ij)}$.

Problem (5.1.1) is minimized parametrically with the tradeoff constant λ which accounts for the tradeoff between minimum norm and minimum misclassification error. Its weight estimate provides an approximate bounding plane for each pairwise comparison. The decision function for problem (5.1.1) is given by:

$$D(x) = sign[x^T w^{(ij)} - \gamma^{(ij)}] = \begin{cases} +1, \text{ if point } (x) \text{ is in class } A^{(i)} \\ -1, \text{ if point } (x) \text{ is in class } A^{(j)} \end{cases} \tag{5.1.5}$$

Now, suppose that in addition to the given data points with their corresponding labels, there is prior knowledge classifying data into one or more categories. We assume that the

knowledge sets in an *n* dimensional space are given in the form of a polyhedral set. Then those are determined by a set of linear equalities and linear inequalities as follows (see Figure 14).

$$B^{(i)}x \le b^{(i)}, \text{ for } x \text{ in } A^{(i)} \text{ and } B^{(j)}x \le b^{(j)}, \text{ for } x \text{ in } A^{(j)}, \text{ or} \tag{5.1.6}$$

or

$$\overline{B}^{(i)}x = \overline{b}^{(i)}, \text{ for } x \text{ in } A^{(i)} \text{ and } \overline{B}^{(j)}x = \overline{b}^{(j)}, \text{ for } x \text{ in } A^{(j)} \tag{5.1.7}$$

All points *x* lying in the polyhedral set defined by (5.1.6) & (5.1.7) belong to class *i*, where $i = 1,..,k$ (*k* classes). Therefore, the following implications must hold for a given ($w^{(ij)}$, $\gamma^{(ij)}$):

$$B^{(i)}x \le b^{(i)} \Rightarrow x^T w^{(ij)} \ge \gamma^{(ij)} + 1 \tag{5.1.8}$$

$$B^{(j)}x \le b^{(j)} \Rightarrow x^T w^{(ij)} \le \gamma^{(ij)} - 1 \tag{5.1.9}$$

$$\overline{B}^{(i)}x = \overline{b}^{(i)} \Rightarrow x^T w^{(ij)} \ge \gamma^{(ij)} + 1 \tag{5.1.10}$$

$$\overline{B}^{(j)}x = \overline{b}^{(j)} \Rightarrow x^T w^{(ij)} \le \gamma^{(ij)} - 1 \tag{5.1.11}$$

Propositions (5.1.8) – (5.1.11) cannot be implemented directly into formulation (5.1.1). Transforming the propositions (5.1.8 – 5.1.11) into constraints which can be incorporated into the $L_M T_R$SVM model can be achieved in two ways. Either applying the nonhomogeneous Farkas theorem of the alternative (Mangasarian, 1994, Fung et al., 2002) or using duality in linear programming (LP) (Bazaraa et al., 1993). The latter is preferred because of its simplicity. It is the dual formulation of the propositions (5.1.8) – (5.1.11) that will be incorporated into the $L_M T_R$SVM model, making the problem feasible.

Fung et al. (2002) developed the propositions for knowledge set classification using Farkas theorem (Mangasarian, 1994).

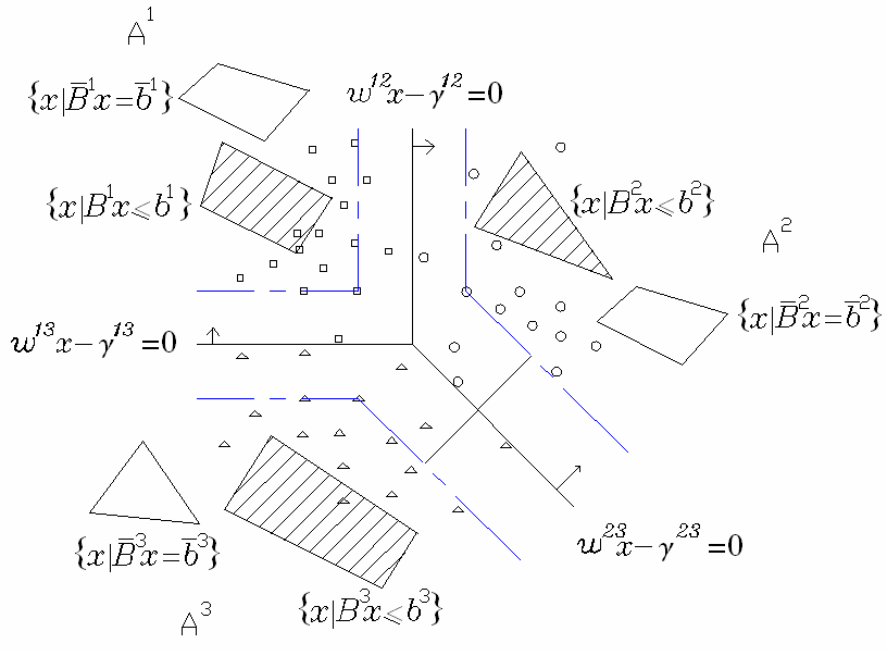

Figure 14: An illustration of a knowledge-based multi-classification with three knowledge sets: belonging in the halfspace of class A^1, belonging in the halfspace of class A^2, and belonging to the halfspace of class A^3.

Proposition 5.1 Knowledge Set Classification (Fung et al., 2002).

The set $\{x \mid B^{(i)}x \le b^{(i)}\}$ *lies in the halfspace* $\{x \mid x^T w^{(ij)} \ge \gamma^{(ij)} + 1\}$ *if and only if, there exists* $u^{(ij)}$ *such that:*

$$\begin{aligned} B^{(i)T}u^{(ij)} + w^{(ij)} &= 0 \\ b^{(i)T}u^{(ij)} + \gamma^{(ij)} + 1 &\le 0 \\ u^{(ij)} \ge 0, \text{ has a solution } &(u^{(ij)}, w^{(ij)}) \end{aligned} \qquad (5.1.12)$$

Proposition 5.1 is the equivalence of implication (5.1.8), and it's the set of linear equations which will readily incorporated into the $L_M T_R$SVM model (5.1.1). (Note that the equivalence can be verified using LP duality (Bazaraa et al., 1993).

5.1.1 Linear Multi-classification Tikhonov Regularization Knowledge-based Support Vector Machine: Incorporation of knowledge set into $L_M T_R$SVM model

Given the knowledge sets:

$$\{x \mid B^{(i)}x \le b^{(i)}\} \text{ or } \{x \mid \overline{B}^{(i)}x = \overline{b}^{(i)}\}, \text{ belonging to class } i, \text{ where} \qquad (5.1.1.1)$$

g_i or $\bar{g}_i$ are the number of prior knowledge (equality or inequality) constraints in class *i.*

$$\text{and } \{x \mid B^{(j)}x \le b^{(j)}\} \text{ or } \{x \mid \overline{B}^{(j)}x = \overline{b}^{(j)}\}, \text{ belonging to class } j, \text{ where} \qquad (5.1.1.2)$$

d_j or $\bar{d}_j$ are the number of prior knowledge (equality or inequality) constraints in class *j,* relative to the bounding planes (5.1.3).

Using Farkas theorem (Mangasarian, 1994), there exist a $u^{(ij)}, v^{(ij)}, \overline{u}^{(ij)}, \overline{v}^{(ij)}$ (using LP Duality Bazaraa et al., 1993) such that:

$$B^{(i)T}u^{(ij)} + w^{(ij)} = 0, \quad b^{(i)T}u^{(ij)} + \gamma^{(ij)} + 1 \le 0, \quad u^{(ij)} \ge 0 \qquad (5.1.1.3)$$

$$B^{(j)T}v^{(ij)} - w^{(ij)} = 0, \quad b^{(j)T}v^{(ij)} - \gamma^{(ij)} + 1 \le 0, \quad v^{(ij)} \ge 0 \qquad (5.1.1.4)$$

or

$$\overline{B}^{(i)T}\overline{u}^{(ij)} + w^{(ij)} = 0, \quad \overline{b}^{(i)T}\overline{u}^{(ij)} + \gamma^{(ij)} + 1 \le 0 \qquad (5.1.1.5)$$

$$\overline{B}^{(j)T}\overline{v}^{(ij)} - w^{(ij)} = 0, \quad \overline{b}^{(j)T}\overline{v}^{(ij)} - \gamma^{(ij)} + 1 \le 0 \qquad (5.1.1.6)$$

Constraints (5.1.1.3) – (5.1.1.6) can be incorporated into the $L_M T_R$SVM formulation (5.1.1) as additional constraints. The resulting optimization problem is as follows:

$$
\begin{aligned}
&\min_{w,\gamma,\xi,\overline{u},\overline{v},(u,v)\geq 0} \ \frac{\lambda}{2}\sum_{i<j}^{k}\left\|w^{(ij)}\right\|^2+\frac{1}{2}\sum_{i<j}^{k}\sum_{t=1}^{m_{(ij)}}(\xi_t^{(ij)})^2\\
&s.t.\ \ \xi_t^{(ij)}=y_t^{(ij)}(\tilde{A}_t^{(ij)}w^{(ij)}-\gamma^{(ij)})-1,\ \ t=1,....,m_{(ij)}\\
&\left.\begin{aligned}
&B^{(i)T}u^{(ij)}+w^{(ij)}=0\\
&b^{(i)T}u^{(ij)}+\gamma^{(ij)}+1\leq 0\\
&B^{(j)T}v^{(ij)}-w^{(ij)}=0\\
&b^{(j)T}v^{(ij)}-\gamma^{(ij)}+1\leq 0
\end{aligned}\right\}\text{priors with inequality sign}\\
&\left.\begin{aligned}
&\overline{B}^{(i)T}\overline{u}^{(ij)}+w^{(ij)}=0\\
&\overline{b}^{(i)T}\overline{u}^{(ij)}+\gamma^{(ij)}+1\leq 0\\
&\overline{B}^{(j)T}\overline{v}^{(ij)}-w^{(ij)}=0\\
&\overline{b}^{(j)T}\overline{v}^{(ij)}-\gamma^{(ij)}+1\leq 0
\end{aligned}\right\}\text{priors with equality sign}\\
&\text{for } i<j,\ \text{where } m_{(ij)}=m_i+m_j,\ \ \tilde{A}^{(ij)}=\begin{pmatrix}A^{(i)}\\A^{(j)}\end{pmatrix}
\end{aligned}
\tag{5.1.1.7}
$$

Problem (5.1.1.7) is a constrained optimization problem, called L_MT_RKSVM. The additional constraints, as they are, assume that each knowledge set lies on the appropriate side of its bounding plane (5.1.3). However, because, we cannot guarantee that such a separating hyperplane exists, error slacks need to be included. Similar to the L_MT_RSVM the error slack will represent the residual of additional constraints, and it will attempt to drive its variables to zero.

For convenience, problem (5.1.1.7) is rewritten in matrix form:

$$
\begin{aligned}
&\min_{w,\gamma,\xi,u,v,r,s,p,\sigma} \ \frac{\lambda}{2}\|w\|^2+\frac{1}{2}\|\xi\|^2+\frac{1}{2}\left[\|r\|^2+\|s\|^2+\|p\|^2+\|\sigma\|^2\right]\\
&s.t.\ \ Aw+E\gamma-e=\xi\\
&\qquad B_u^Tu+I_uw=r\\
&\qquad B_{bu}^Tu+E_u\gamma+e_u=p\\
&\qquad B_v^Tv-I_vw=s\\
&\qquad B_{bv}^Tv-E_v\gamma+e_v=\sigma
\end{aligned}
\tag{5.1.1.8}
$$

Note that weights can be assigned to each term of the objective function, constructing a multi-objective optimization (MOP) problem with 6 objectives.

All knowledge sets have a unique set of Lagrange multipliers, where $u=[u^{(ij)T},....,u^{(k-1)kT},\overline{u}^{(ij)T},....,\overline{u}^{(k-1)kT}]^T$ ($u\in R^{g\times 1}$) is a vector of all multipliers referring to the i^{th} class, and $v=[v^{(ij)T},....,v^{(k-1)kT},\overline{v}^{(ij)T},....,\overline{v}^{(k-1)kT}]^T$ ($v\in R^{d\times 1}$) is a vector of all multipliers referring to the j^{th} class. ξ is a residual error vector accounting for the training data error, and vectors r,s,p,σ are residual error vectors accounting for the knowledge set error (i.e., violation of the knowledge constraints).

The following matrices are defined for all $i<j,\ \ i,j=1,....k$ classes (Note that matrices (5.1.1.9) – (5.1.1.12) describe the formation of the prior knowledge):

Matrix B_u (B_v) are diagonal block matrices whose diagonals contain knowledge sets belonging to the i^{th} (or j^{th}) class. The i^{th} (or j^{th}) knowledge set is derived from the inequality and equality prior knowledge constraints. The diagonals of B_u (B_v) are $B_{ij}^{(i)}\in R^{g_i\times n}$, $\overline{B}_{ij}^{(i)}\in R^{\overline{g}_i\times n}$ ($B_{ij}^{(j)}\in R^{d_j\times n}$, $\overline{B}_{ij}^{(j)}\in R^{\overline{d}_j\times n}$). Matrix B_{bu} (B_{bv}) is a matrix consisting of vectors $b_{(ij)}^{(i)}\in R^{g_i\times 1}$, $\overline{b}_{(ij)}^{(i)}\in R^{\overline{g}_i\times 1}$ ($b_{(ij)}^{(j)}\in R^{d_j\times 1}$, $\overline{b}_{(ij)}^{(j)}\in R^{\overline{d}_j\times 1}$) and each component of the vectors is the bound of the knowledge set. Matrix I_u (I_v) is a block matrix consisting of identity matrices $I_{u(ij)},\overline{I}_{u(ij)}\in R^{n\times n}$ ($I_{v(ij)},\overline{I}_{v(ij)}\in R^{n\times n}$). Matrix E_u (E_v) is a matrix consisting of entries $e_{u(ij)},\overline{e}_{u(ij)}\in R$ ($e_{v(ij)},\overline{e}_{v(ij)}\in R$), where $e_{u(ij)},\overline{e}_{u(ij)},e_{v(ij)},\overline{e}_{v(ij)}$ are equal to one. Vector e_u (e_v) is a vector of ones, where each entry corresponds to a vector $b_{(ij)}^{(i)}\in R^{g_i\times 1}$, $\overline{b}_{(ij)}^{(i)}\in R^{\overline{g}_i\times 1}$ ($b_{(ij)}^{(j)}\in R^{d_j\times 1}$, $\overline{b}_{(ij)}^{(j)}\in R^{\overline{d}_j\times 1}$). Each i^{th} and j^{th} entry into matrices and vector B_u, B_{bu}, I_u, E_u, e_u,

and B_v , B_{bv} , I_v , E_v , e_v completes a pairwise comparison. Matrices A and E and vector e are defined in (4.1.7) respectively.

The formulation is applicable to $k \geq 2$, and can be expressed in the following matrix form:

$$B_{u_{g\times 2n\cdot\bar{n}}} = \begin{bmatrix} B_{(ij)}^{(i)} & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & 0 & \cdots & \cdots & \vdots \\ \vdots & 0 & B_{(k-1)k}^{(k-1)} & 0 & \vdots & \vdots \\ \vdots & \vdots & 0 & \bar{B}_{(ij)}^{(i)} & 0 & \vdots \\ \vdots & \vdots & \vdots & 0 & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & \bar{B}_{(k-1)k}^{(k-1)} \end{bmatrix}, I_{u_{2n\cdot\bar{n}\times n\cdot\bar{n}}} = \begin{bmatrix} I_{u(ij)} & 0 & 0 \\ 0 & \ddots & \vdots \\ \vdots & 0 & I_{u(k-1)k} \\ \bar{I}_{u(ij)} & 0 & 0 \\ 0 & \ddots & \vdots \\ \vdots & 0 & \bar{I}_{u(k-1)k} \end{bmatrix} \quad (5.1.1.9)$$

where $g = \sum_{i<j}^{k} (g_i + \bar{g}_i)_{ij}$

$$B_{v_{d\times 2n\cdot\bar{n}}} = \begin{bmatrix} B_{ij}^{(j)} & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & 0 & \cdots & \cdots & \vdots \\ \vdots & 0 & B_{(k-1)k}^{(k)} & 0 & \vdots & \vdots \\ \vdots & \vdots & 0 & \bar{B}_{ij}^{(j)} & 0 & \vdots \\ \vdots & \vdots & \vdots & 0 & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & \bar{B}_{(k-1)k}^{(k)} \end{bmatrix}, I_{v_{2n\cdot\bar{n}\times n\cdot\bar{n}}} = \begin{bmatrix} I_{v(ij)} & 0 & 0 \\ 0 & \ddots & \vdots \\ \vdots & 0 & I_{v(k-1)k} \\ \bar{I}_{v(ij)} & 0 & 0 \\ 0 & \ddots & \vdots \\ \vdots & 0 & \bar{I}_{v(k-1)k} \end{bmatrix} \quad (5.1.1.10)$$

where $d = \sum_{i<j}^{k} (d_j + \bar{d}_j)_{ij}$

$$B_{bu_{g\times 2\cdot\bar{n}}} = \begin{bmatrix} b_{(ij)}^{(i)} & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & 0 & \cdots & \cdots & \vdots \\ \vdots & 0 & b_{(k-1)k}^{(k-1)} & 0 & \vdots & \vdots \\ \vdots & \vdots & 0 & \bar{b}_{(ij)}^{(i)} & 0 & \vdots \\ \vdots & \vdots & \vdots & 0 & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & \bar{b}_{(k-1)k}^{(k-1)} \end{bmatrix}, E_{u_{2\bar{n}\times\bar{n}}} = \begin{bmatrix} e_{u(ij)} & 0 & 0 \\ 0 & \ddots & \vdots \\ \vdots & 0 & e_{u(k-1)k} \\ \bar{e}_{u(ij)} & 0 & 0 \\ 0 & \ddots & \vdots \\ \vdots & 0 & \bar{e}_{u(k-1)k} \end{bmatrix}, e_{u_{2\bar{n}\times 1}} = \begin{pmatrix} e_{u(ij)} \\ \vdots \\ e_{u(k-1)k} \\ \bar{e}_{u(ij)} \\ \vdots \\ \bar{e}_{u(k-1)k} \end{pmatrix} \quad (5.1.1.11)$$

$$B_{bv_{d\times 2\bar{n}}}=\begin{bmatrix} b_{(ij)}^{(j)} & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & 0 & \cdots & \cdots & \vdots \\ \vdots & 0 & b_{(k-1)k}^{(k)} & 0 & \vdots & \vdots \\ \vdots & \vdots & 0 & \bar{b}_{(ij)}^{(j)} & 0 & \vdots \\ \vdots & \vdots & \vdots & 0 & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & \bar{b}_{(k-1)k}^{(k)} \end{bmatrix}, E_{v_{2\bar{n}\times\bar{n}}}=\begin{bmatrix} e_{v(ij)} & 0 & 0 \\ 0 & \ddots & \vdots \\ \vdots & 0 & e_{v(k-1)k} \\ \bar{e}_{v(ij)} & 0 & 0 \\ 0 & \ddots & \vdots \\ \vdots & 0 & \bar{e}_{v(k-1)k} \end{bmatrix}, e_{v_{2\bar{n}\times 1}}=\begin{pmatrix} e_{v(ij)} \\ \vdots \\ e_{v(k-1)k} \\ \bar{e}_{v(ij)} \\ \vdots \\ \bar{e}_{v(k-1)k} \end{pmatrix} \quad (5.1.1.12)$$

To eliminate the error slack ξ from constraints, problem (5.1.1.8) can be rewritten as:

$$\min_{w,\gamma,u,v} f_{L_M T_R KSVM}(w,\gamma,u,v)=\begin{bmatrix} \frac{\lambda}{2}\|w\|^2+\frac{1}{2}\|Aw+E\gamma-e\|^2+ \\ \frac{1}{2}\left[\left\|B_u^T u+I_u w\right\|^2+\left\|B_v^T v-I_v w\right\|^2\right]+ \\ \frac{1}{2}\left[\left\|B_{bu}^T u+E_u\gamma+e_u\right\|^2+\left\|B_{bv}^T v-E_v\gamma+e_v\right\|^2\right] \end{bmatrix} \quad (5.1.1.13)$$

Problem (5.1.1.13) is the final model which incorporates the knowledge sets (as defined in (5.1.1.1) and (5.1.1.2)) into the $L_M T_R$SVM model. The tradeoff constant λ is also run through a series, or range, of values to achieve the best result. If its value increases then the minimum norm is achieved, but at the expense of having a higher training residual error and higher knowledge set residual error. If data isn't available, the second term of the $L_M T_R$KSVM model can be dropped. This result in classifiers based strictly on knowledge sets, and it is useful for situations of which only expert knowledge exists.

The L_2 norm of all terms is considered because of its strong convexity of the objective function. Nonetheless, the L_1 norm can also be used, by changing problem (5.1.1.8) into a linear programming (LP) problem. However, explicit expressions to the solution of the classification weights cannot be obtained in the LP solution. Problem (5.1.1.13) is a convex unconstrained optimization problem which has a unique minimum point. To find its minimum point, the optimality conditions need to be written explicitly.

Expanding each term in problem (5.1.1.13):

$$\frac{\lambda}{2}\|w\|^2 = \frac{\lambda}{2} w^T w \tag{5.1.1.14}$$

$$\begin{aligned}\frac{1}{2}\|Aw + E\gamma - e\|^2 &= \frac{1}{2}\left[Aw + E\gamma - e\right]^T \left[Aw + E\gamma - e\right] \\ &= \frac{1}{2}\left[w^T A^T Aw + 2w^T A^T E\gamma - 2e^T Aw - 2\gamma^T E^T e + \gamma^T E^T E\gamma + e^T e\right]\end{aligned} \tag{5.1.1.15}$$

$$\begin{aligned}\frac{1}{2}\left\|B_u^T u + I_u w\right\|^2 &= \frac{1}{2}\left[B_u^T u + I_u w\right]^T \left[B_u^T u + I_u w\right] \\ &= \frac{1}{2}\left[u^T B_u B_u^T u + 2u^T B_u I_u w + w^T {I_u}^T I_u w\right]\end{aligned} \tag{5.1.1.16}$$

$$\begin{aligned}\frac{1}{2}\left\|B_v^T v - I_v w\right\|^2 &= \frac{1}{2}\left[B_v^T v - I_v w\right]^T \left[B_v^T v - I_v w\right] \\ &= \frac{1}{2}\left[v^T B_v B_v^T v - 2v^T B_v I_v w + w^T {I_v}^T I_v w\right]\end{aligned} \tag{5.1.1.17}$$

$$\begin{aligned}\frac{1}{2}\left\|B_{bu}^T + E_u\gamma + e_u\right\|^2 &= \frac{1}{2}\left[B_{bu}^T u + E_u\gamma + e_u\right]^T \left[B_{bu}^T u + E_u\gamma + e_u\right] \\ &= \frac{1}{2}\begin{bmatrix} u^T B_{bu} B_{bu}^T u + 2u^T B_{bu} E_u\gamma + \\ 2u^T B_{bu} e_u + 2{e_u}^T E_u\gamma + \gamma^T {E_u}^T E_u\gamma + {e_u}^T e_u \end{bmatrix}\end{aligned} \tag{5.1.1.18}$$

$$\begin{aligned}\frac{1}{2}\left\|B_{bv}^T v - E_v\gamma + e_v\right\|^2 &= \frac{1}{2}\left[B_{bv}^T v - E_v\gamma + e_v\right]^T \left[B_{bv}^T v - E_v\gamma + e_v\right] \\ &= \frac{1}{2}\begin{bmatrix} v^T B_{bv} B_{bv}^T v - 2v^T B_{bv} E_v\gamma + \\ 2v^T B_{bv} e_v - 2{e_v}^T E_v\gamma + \gamma^T {E_v}^T E_v\gamma + {e_v}^T e_v \end{bmatrix}\end{aligned} \tag{5.1.1.19}$$

To find the minimizing solution to problem (5.1.1.13) we need to write its optimality conditions. Below are the optimality conditions for $f_{L_M T_R KSVM}(w,\gamma,u,v)$, obtained by setting the partial derivatives of (5.1.1.13) with respect to w, u, v and γ equal to zero:

$$\frac{df}{dw} = \lambda w + A^T Aw + A^T E\gamma - A^T e + {I_u}^T B_u^T u + {I_u}^T I_u w - {I_v}^T B_v^T v + {I_v}^T I_v w = 0 \tag{5.1.1.20}$$

$$\frac{df}{du} = B_u B_u^T u + B_u I_u w + B_{bu} B_{bu}^T u + B_{bu} e_u \gamma + B_{bu} e_u = 0 \tag{5.1.1.21}$$

$$\frac{df}{dv} = B_v B_v^T v - B_v I_v w + B_{bv} B_{bv}^T v - B_{bv} e_v \gamma + B_{bv} e_v = 0 \tag{5.1.1.22}$$

$$\frac{df}{d\gamma} = \begin{bmatrix} E^T A w - E^T e + E^T E \gamma + E_u{}^T B_{bu}^T u + \\ E_u{}^T e_u + E_u{}^T E_u \gamma - E_v{}^T B_{bv}^T v - E_v{}^T e_v + E_v{}^T E_v \gamma \end{bmatrix} = 0 \tag{5.1.1.23}$$

To provide explicit matrix expressions for w and γ, the optimality conditions need to be rearranged. From equation (5.1.1.21) and (5.1.1.22) the following expressions are obtained for variables u and v:

$$\begin{aligned} &u = -UB_u I_u w - UB_{bu} E_u \gamma - UB_{bu} e_u, \\ &\text{where } U_{g\times g} = \left(B_u B_u^T + B_{bu} B_{bu}^T \right)^{-1} \end{aligned} \tag{5.1.1.24}$$

$$\begin{aligned} &v = VB_v I_v w + VB_{bv} E_v \gamma - VB_{bv} e_v, \\ &\text{where } V_{d\times d} = \left(B_v B_v^T + B_{bv} B_{bv}^T \right)^{-1} \end{aligned} \tag{5.1.1.25}$$

Now that estimates are provided for *u* and *v*, the location (bias) of each pairwise hyperplane relative to the origin can be estimated by substituting (5.1.1.24) and (5.1.1.25) in equation (5.1.1.23):

$$\begin{aligned} &\gamma = H(-Lw + a) \\ &\text{where } \gamma = [\gamma^{(ij)}, ..., \gamma^{(k-1)k}]^T, \text{ for } i < j \end{aligned} \tag{5.1.1.26}$$

$$H_{\bar{n}\times\bar{n}} = \left[E^T E + E_u{}^T \left[E_u - B_{bu}^T UB_{bu} E_u \right] + E_v{}^T \left[E_v - B_{bv}^T VB_{bv} E_v \right] \right]^{-1} \tag{5.1.1.27}$$

$$\begin{aligned} L_{\bar{n}\times n\cdot\bar{n}} &= \left[E^T A - E_u{}^T B_{bu}^T UB_u I_u - E_v{}^T B_{bv}^T VB_v I_v \right] \\ a_{\bar{n}\times 1} &= \left[E^T e - E_u{}^T \left[e_u - B_{bu}{}^T UB_{bu} e_u \right] + E_v{}^T \left[e_v - B_{bv}{}^T VB_{bv} e_v \right] \right] \end{aligned} \tag{5.1.1.28}$$

With explicit solutions to u, v and γ, already determined under the assumption that *U*, *V* and *H* are invertible matrices, when substituted into equation (5.1.1.20), the solution to the normal vector w can conveniently be expressed as:

$$\begin{aligned} &w = M^{-1} z \\ &\text{where } w = [w^{(ij)T}, ..., w^{(k-1)kT}]^T, \text{ for } i < j \end{aligned} \tag{5.1.1.29}$$

The decision function for classifying a point is given by:

$$D(x) = sign[x^T w^{(ij)} - \gamma^{(ij)}] = \begin{cases} +1, & \text{if point } (x) \text{ is in class } A^{(i)} \\ -1, & \text{if point } (x) \text{ is in class } A^{(j)} \end{cases} \tag{5.1.1.30}$$

To classify a new point x, the voting approach strategy (Santosa, et al., 2002; Hsu & Lin, 2002) is employed. For example, if sign $[\, x^T w^{(ij)} - \gamma^{(ij)} \,]$ says that x is in the i^{th} class, then the vote for the i^{th} class is increased by one. Otherwise, the j^{th} is increased by one. Then we predict x as being in the class with the largest vote. In the case that those two classes have identical votes, select the one with the smallest index.

The corresponding linear system to (5.1.1.29) is:

$$Mw = z\,, \tag{5.1.1.31}$$

where

$$M_{n\cdot\bar{n}\times n\cdot\bar{n}} = \begin{bmatrix} \lambda I + A^T \left[A - EHL \right] + \\ I_u{}^T \left[I_u - B_u{}^T U \left[B_u I_u - B_{bu} E_u HL \right] \right] + \\ I_v{}^T \left[I_v - B_v{}^T V \left[B_v I_v - B_{bv} E_v HL \right] \right] \end{bmatrix} \tag{5.1.1.32}$$

$$z_{n\cdot\bar{n}\times 1} = \begin{bmatrix} A^T \left[e - EHa \right] + \\ I_u{}^T B_u{}^T U B_{bu} \left[e_u + E_u Ha \right] - \\ I_v{}^T B_v{}^T V B_{bv} \left[e_v - E_v Ha \right] \end{bmatrix} \tag{5.1.1.33}$$

(5.1.1.29) provides a solution to the linear system of equations in (5.1.1.31). Methods for solving the linear system include matrix decomposition methods, or iterative based methods (Lewis et al., 2006; Bazaraa et al., 1993). Its solution involves the inversion of a smaller dimensional matrix (M) of the magnitude $(n*\bar{n})\times(n*\bar{n})$, where $\bar{n} = k(k-1)/2$. Below is an algorithm for $L_M T_R$KSVM.

Algorithm 5.1 Linear Multi-classification Tikhonov Regularization Knowledge-based SVM ($k \geq 2$): Given a data set in R^n that are represented by a matrix $A^{(i)} \in R^{m_i \times n}$, where $i = 1,..,k$ (k classes), and knowledge sets $\{x \mid B^{(i)}x \leq b^{(i)}\}$ or $\{x \mid \overline{B}^{(i)}x = \overline{b}^{(i)}\}$ belonging to $A^{(i)}$, $\{x \mid B^{(j)}x \leq b^{(j)}\}$ or $\{x \mid \overline{B}^{(j)}x = \overline{b}^{(j)}\}$ belonging to $A^{(j)}$. Let $A^{(i)}$ be a $m_i \times n$ matrix whose rows are points in the i^{th} class. Let $A^{(j)}$ be a $m_j \times n$ matrix whose rows are points in the j^{th} class. Computing weights w and γ for linear classifier (5.1.1.30) is as follows.

$L_M T_R$KSVM Algorithm:

Step 1: Compute M from (5.1.1.32) for a specific constant λ.

Step 2: Compute z from (5.1.1.33).

Step 3: Determine w from (5.1.1.29).

Step 4: Determine γ from (5.1.1.26).

Step 5: Classify a new point x by (5.1.1.30).

It is in step 3 that matrix methods or iterative methods are invoked to obtain the solution to w.

5.1.2 Numerical Example (AND & three class problem)

To illustrate the workings of the proposed method, two small problems are considered; the AND problem and a simple three class problem, both with prior knowledge. The computations for the two small problems were conducted using MATLAB. Next we describe the AND Problem with the following data matrices and knowledge sets:

$$A^{(1)} = \begin{pmatrix} 1 & 1 \end{pmatrix}, \; A^{(2)} = \begin{pmatrix} -1 & -1 \\ -1 & 1 \\ 1 & -1 \end{pmatrix} \tag{5.1.2.1}$$

$$\{x \in R^2 \mid x_1 \geq 0.5,\ x_2 \geq 0.5\} \Leftrightarrow \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} \Leftrightarrow B^{(1)}x \leq b^{(1)} \qquad (5.1.2.2)$$

$$\{x \in R^2 \mid x_1 \leq -0.5,\ x_2 \leq -0.5\} \Leftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} \Leftrightarrow B^{(2)}x \leq b^{(2)} \qquad (5.1.2.3)$$

Using the matrix definitions in (4.1.7) and (5.1.1.9) – (5.1.1.12), the data and prior knowledge sets given in (5.1.2.1) – (5.1.2.3) can be represented as:

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix},\ E = \begin{pmatrix} -1 \\ 1 \\ 1 \\ 1 \end{pmatrix},\ B_u = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix},\ B_v = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad (5.1.2.4)$$

$$I_u = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},\ I_v = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},\ b_u = b_v = \begin{pmatrix} -0.5 \\ -0.5 \end{pmatrix},\ E_u = E_v = 1 \qquad (5.1.2.5)$$

Problem (5.1.1.13) is solved with the given data sets in (5.1.2.1) and knowledge sets in (5.1.2.2 & 5.1.2.3) defined in the matrices (5.1.2.4) & (5.1.2.5). As the tradeoff increases, the approximate margin (L_2 norm of w) increases (decreases). From Table 6 and Figure 15, it is evident that a specific constant corresponds to a specific margin, and it is within this approximate margin that the sum of square errors is minimized. At λ =0, the margin is smallest; at that juncture only the sum of square errors of the sets is minimized. When λ =1, both the L_2 norm of w and the sum of square errors of the data sets are minimized. At λ=3, 4 & 5, the model can over generalize because the margin becomes too large. Illustrations of the approximate classifiers are shown in Figure 15. Note that at λ =3, 4 & 5 (yellow, black & blue dash line) the classifiers are over forecasting (solution to the norm is too small).

Since the supporting hyperplanes are approximate bounding planes for classes *i* and *j*, some errors are tolerable. Also, there are no support vectors as a result of the

approximation because no points lie on the supporting hyperplanes (see Figure 16). However, the approximate margin does increase as a direct consequence of the approximate bounding planes centered in the region of its categorical points, but pushed as far apart from each class (see Figure 16).

Optimal separating hyperplane for λ =0 (blue line Figures 16 & 17):

$$w^T x - \gamma = 0 \Rightarrow (0.5714 \quad 0.5714)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 0.3750 \Rightarrow 0.5714x_1 + 0.5714x_2 = 0.3750 \quad (5.1.2.6)$$

The linear programming formulation for the knowledge based SVM (KBSVM) by Fung et al. (2002) was also trained on the AND problem, and the optimal separating hyperplane is given through the green line of Figure 17:

$$w^T x - \gamma = 0 \Rightarrow \begin{pmatrix} w_1 & w_2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \gamma = (2 \quad 2)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 1 \Rightarrow 2x_1 + 2x_2 = 1 \quad (5.1.2.7)$$

Table 6. $L_M T_R$KSVM solution to AND Problem with knowledge sets (varying tradeoff constant λ)

λ	w_1	w_2	γ	Margin
0	0.5714	0.5714	0.3750	1.2374
0.1	0.5594	0.5594	0.3750	1.2640
0.5	0.5161	0.5161	0.3750	1.3700
1	0.4706	0.4706	0.3750	1.5026
2	0.4	0.4	0.3750	1.7678
3	0.3478	0.3478	0.3750	2.0329
4	0.3077	0.3077	0.3750	2.2981
5	0.2759	0.2759	0.3750	2.5633
KBSVM	2	2	1	0.3536

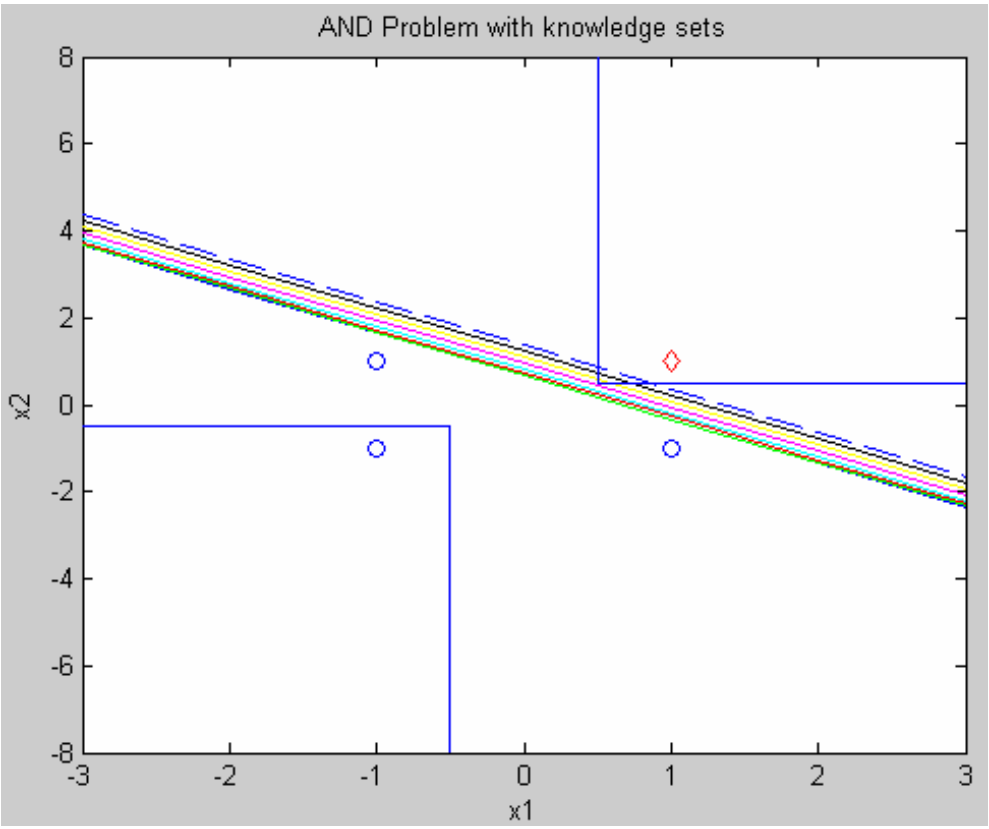

Figure 15: Illustration of L_MT_RKSVM optimal hyperplanes with varying tradeoff constants for AND problem. Knowledge sets belonging to class 1 and 2 are intersecting solid blues. The blue line for λ =0, green line for λ =0.1, red line for λ =0.5, cyan line for λ =1, magenta line for λ =2, yellow line for λ =3, black line for λ =4, blue dash line for λ =5. Red diamond for class A^1, blue circles for class A^2.

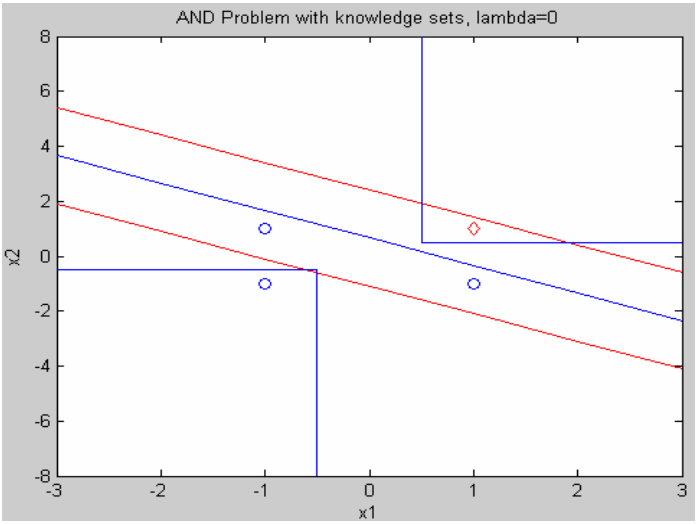

Figure 16: Illustration of L_MT_RKSVM hyperplanes λ=0 for AND problem. Knowledge sets belonging to class 1 and 2 are intersecting solid blues lines. The supporting approximate hyperplanes are in red, the optimal hyperplane in blue. Red diamond for class A^1, blue circles for class A^2.

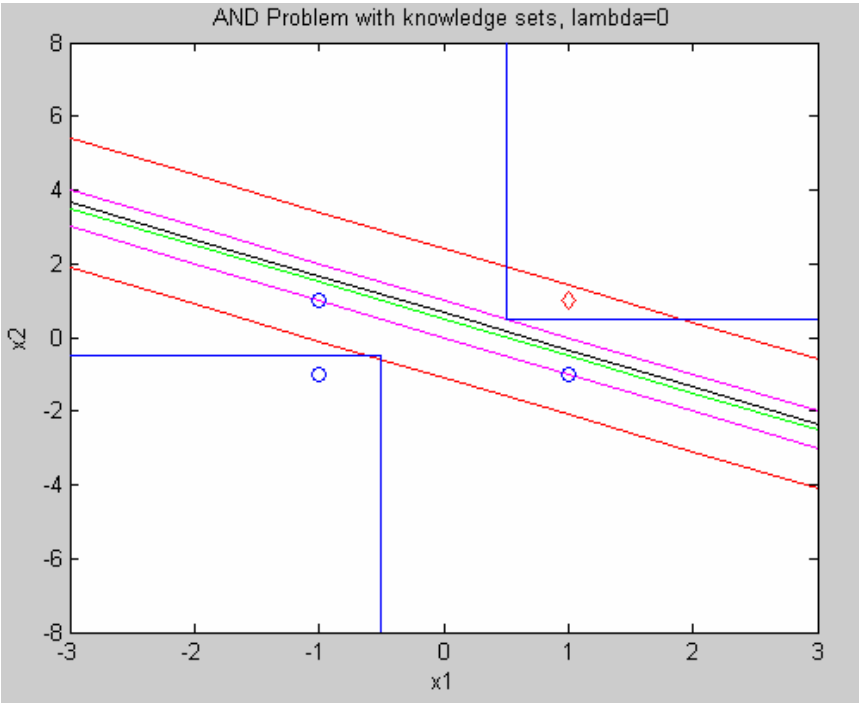

Figure 17: Illustration of $L_M T_R$KSVM hyperplanes λ=0 and KBSVM for AND problem. Knowledge sets belonging to class 1 and 2 are intersecting solid blues lines. Red diamond for class A^1, blue circles for class A^2. For $L_M T_R$KSVM model, the supporting approximate hyperplanes are in red, the optimal hyperplane in black. For KBSVM model, the supporting approximate hyperplanes are in magenta, the optimal hyperplane in green.

Next we describe the three class problem, given the data sets and knowledge sets below

(see Figure 18):

$$A^{(1)} = \begin{pmatrix} -1 & 1 \end{pmatrix},\ A^{(2)} = \begin{pmatrix} 1 & 1 \end{pmatrix},\ A^{(3)} = \begin{pmatrix} -1 & -1 \end{pmatrix} \tag{5.1.2.8}$$

$$\{x \in R^2 \mid x_1 \le -0.5,\ x_2 \ge 0.5\} \Leftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} \Leftrightarrow B^{(1)} x \le b^{(1)} \tag{5.1.2.9}$$

$$\{x \in R^2 \mid x_1 \ge 0.5,\ x_2 \ge 0.5\} \Leftrightarrow \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} \Leftrightarrow B^{(2)} x \le b^{(2)} \tag{5.1.2.10}$$

$$\{x \in R^2 \mid x_1 \le -0.5,\ x_2 \le -0.5\} \Leftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} \Leftrightarrow B^{(3)} x \le b^{(3)} \tag{5.1.2.11}$$

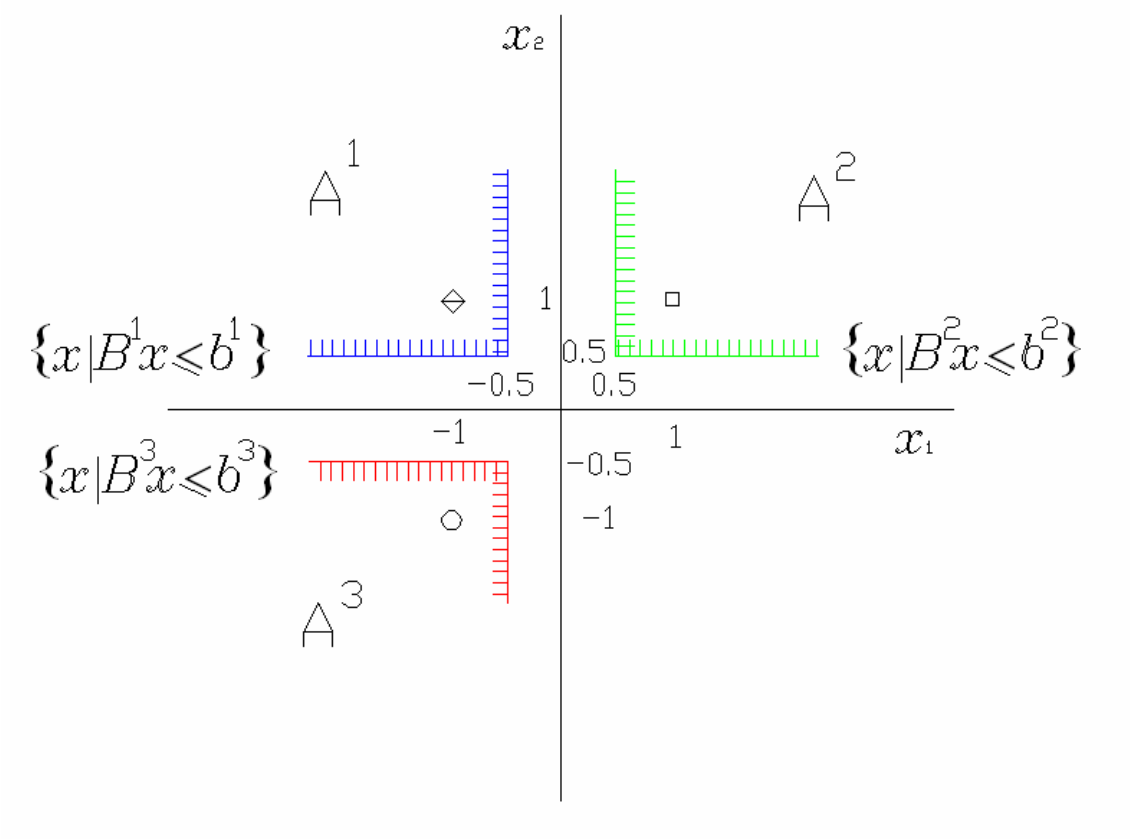

Figure 18: An illustration of a knowledge-based multi-class SVM with three knowledge sets; The set in blue are knowledge sets belonging in the halfspace of class A^1 (diamond), in green are knowledge sets belonging in the halfspace of class A^2 (square), and in red are knowledge sets belonging in the halfspace of class A^3 (circle).

Problem (5.1.1.13) is solved on the above simple three class problem with prior knowledge. All the experiments for this example were performed using MATLAB.

Table 7 summarizes the classification weight statistics. Notice that at λ =0, there is no solution (infeasible), this will happen because at 0, M in (5.1.1.32) is a singular matrix.

The tradeoff constant not only tries to find a balance between the minimum norm and error, it also forces matrix M to be nonsingular (M is of full rank). Hence, for this problem the tradeoff constant should be greater than zero ($\lambda > 0$).

Optimal separating hyperplanes for λ =0.0001:

$$\text{class 1 v 2} \Rightarrow \begin{pmatrix} w_1 & w_2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \gamma = (-1.1428 \quad 0)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 0 = -1.1428x_1 = 0$$

$$\text{class 1 v 3} \Rightarrow \begin{pmatrix} w_1 & w_2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \gamma = (0 \quad 1.1428)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 0 = 1.1428x_2 = 0 \qquad (5.1.2.12)$$

$$\text{class 2 v 3} \Rightarrow \begin{pmatrix} w_1 & w_2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \gamma = (0.5714 \quad 0.5714)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 0 = 0.5714x_1 + 0.5714x_2 = 0$$

Table 7. $L_M T_R$KSVM solution to THREE class Problem (varying λ)

	class 1 v 2				class 1 v 3				class 2 v 3			
λ	w_1	w_2	γ	margin	w_1	w_2	γ	margin	w_1	w_2	γ	margin
0	Infeasible											
0.0001	-1.1428	0	0	0.875	0	1.1428	0	0.875	0.5714	0.5714	0	1.2375
0.0005	-1.1426	0	0	0.8752	0	1.1426	0	0.8752	0.5714	0.5714	0	1.2375
0.001	-1.1424	0	0	0.8754	0	1.1424	0	0.8754	0.5713	0.5713	0	1.2377
0.005	-1.1404	0	0	0.8769	0	1.1404	0	0.8769	0.5708	0.5708	0	1.2388
0.01	-1.138	0	0	0.8787	0	1.138	0	0.8787	0.5702	0.5702	0	1.2401
0.05	-1.1189	0	0	0.8937	0	1.1189	0	0.8937	0.5654	0.5654	0	1.2507
0.1	-1.0959	0	0	0.9125	0	1.0959	0	0.9125	0.5594	0.5594	0	1.264
0.5	-0.9412	0	0	1.0625	0	0.9412	0	1.0625	0.5161	0.5161	0	1.37
1	-0.8	0	0	1.25	0	0.8	0	1.25	0.4706	0.4706	0	1.5026
5	-0.3636	0	0	2.75	0	0.3636	0	2.75	0.2759	0.2759	0	2.5633

The optimal separating hyperplane are the same, however, since the pairwise comparisons are distinctive and bounded by a ± 1, the supporting hyperplanes cannot be the same. As the tradeoff increases, the approximate margin (L_2 norm of w) increases (decreases). At small increments of λ, the classifiers are not significantly different and the approximate margins are much closer. Values of λ between 0.0001 and 0.01 give similar approximate hyperplanes and minimize the errors. At larger increments of λ, the influence on the classifiers becomes obviously large. Their corresponding approximate bounding planes move further away from the optimal hyperplane, data points, and knowledge sets belonging to its respective class (see Figures 19, 20 & 21).

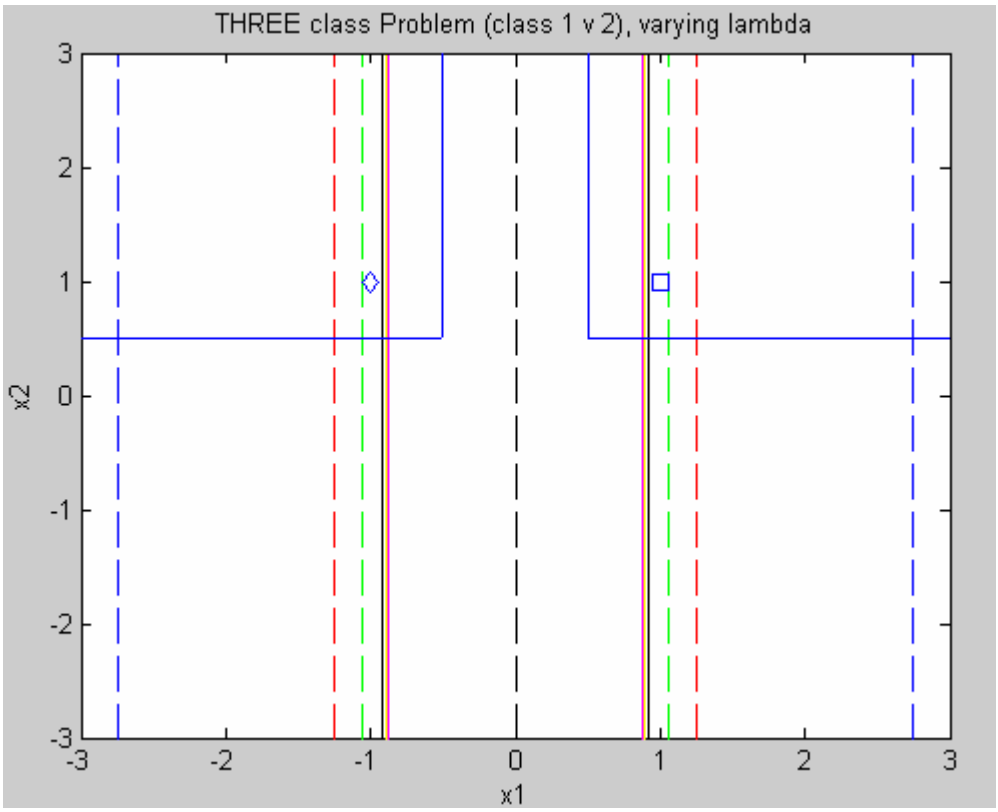

Figure 19: Illustration of $L_M T_R$KSVM hyperplane with varying tradeoff constants for THREE class problem. Knowledge sets belonging in class 1 and 2 are intersecting solid blues, optimal hyperplane in black dashed line, supporting hyperplanes red line (λ =0.0001), blue line (λ =0.0005), green line (λ =0.001), cyan line (λ =0.005), magenta line (λ =0.01), yellow line (λ =0.05), black line (λ =0.1), green dash line (λ =0.5), red dash line (λ =1), blue dash line (λ =5), diamond for class A^1, square for class A^2.

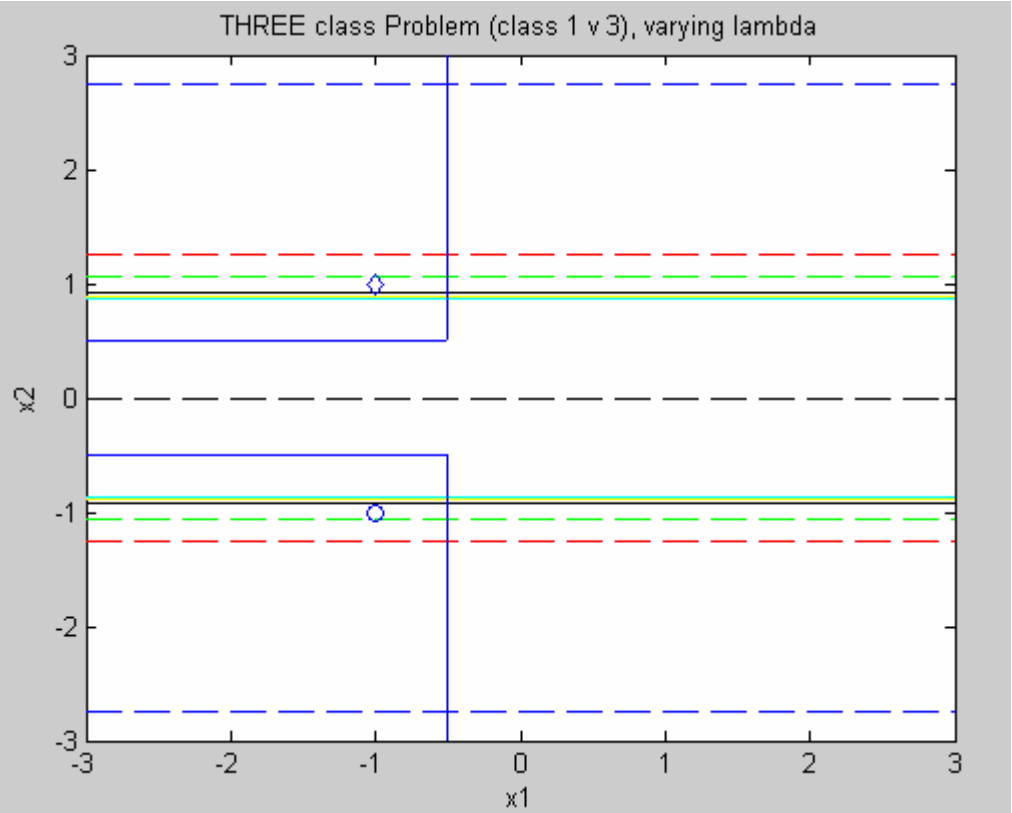

Figure 20: Illustration of $L_M T_R$KSVM hyperplane with varying tradeoff constants for THREE class problem. Knowledge sets belonging in class 1 and 3 are intersecting solid blues, optimal hyperplane in black dashed line, supporting hyperplanes red line (λ =0.0001), blue line (λ =0.0005), green line (λ =0.001), cyan line (λ =0.005), magenta line (λ =0.01), yellow line (λ =0.05), black line (λ =0.1), green dash line (λ =0.5), red dash line (λ =1), blue dash line (λ =5), diamond for class A^1, circle for class A^3.

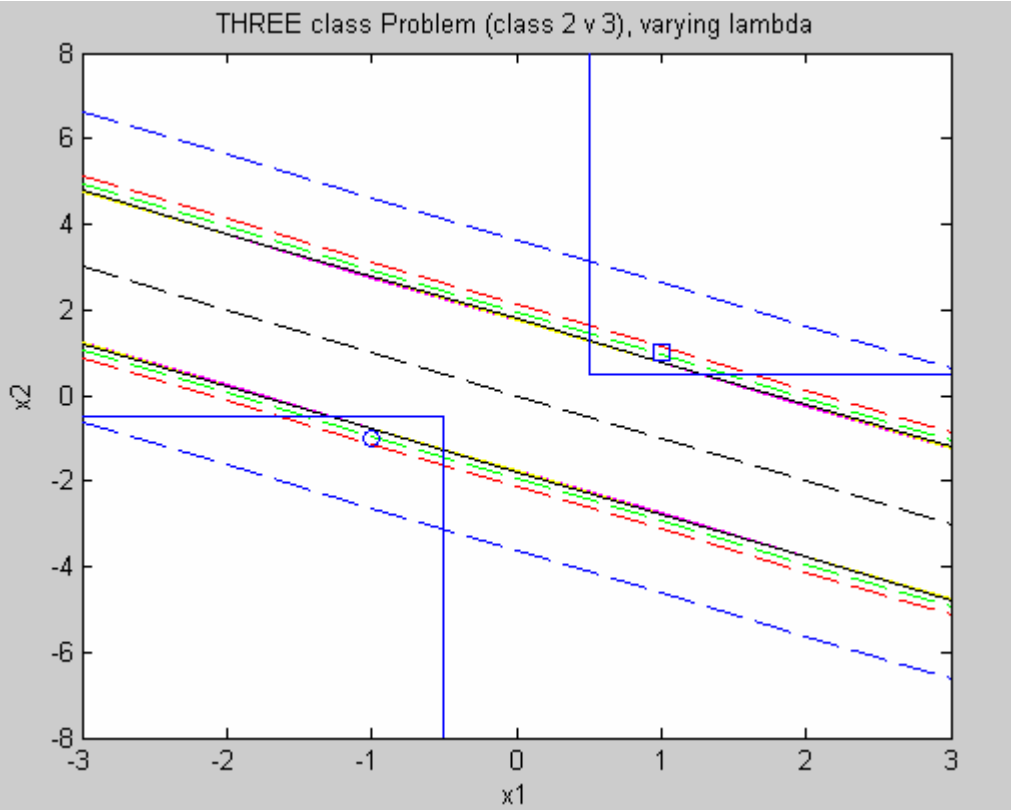

Figure 21: Illustration of $L_M T_R$KSVM hyperplane with varying tradeoff constants for THREE class problem. Knowledge sets belonging in class 2 and 3 are intersecting solid blues, optimal hyperplane in black dashed line, supporting hyperplanes red line (λ =0.0001 & 0.0005), blue line (λ =0.001), cyan line (λ =0.005), magenta line (λ =0.01), yellow line (λ =0.05), black line (λ =0.1), green dash line (λ =0.5), red dash line (λ =1), blue dash line (λ =5), square for class A^2, circle for class A^3.

In Figures 19, 20 & 21, the black dash line is the optimal hyperplane—line perpendicular to the normal vector w—for separation between classes 1 & 2, classes 1 & 3, and classes 2 & 3 respectively. Since the optimal hyperplanes are the same irrespective of the tradeoff constant for a specific pairwise comparison, only the optimal classifier with λ =0.0001 is depicted on Figures 19, 20 & 21 together with the bounding planes for the varying tradeoff constants. As the tradeoff increases, the approximate bounding planes move far apart from the optimal hyperplane and points unique to the bounding planes.

Similar to the $L_M T_R$SVM model, it happens that at large values the approximate margin becomes too wide, thereby increasing the error margin of the point from its position to its respective approximate bounding plane.

5.2 Nonlinear Multi-classification Tikhonov Regularization Knowledge-based Kernel Machine

Similar to chapter 4, section 4.2, to obtain nonlinear classifiers it is essential to carry out a nonlinear mapping from the input space to a feature space (Burges, 1998; Cristianini & Shawe-Taylor, 2000) using a mapping function (see Fig. 3),

$$\phi : R^n \rightarrow F \tag{5.2.1}$$

Replacing the primal variable *w* by its equivalent dual representation below:

$$w = \overline{A}^T Y \alpha \tag{5.2.2}$$

Then problem (5.1.1.13) can be modified as follows:

$$\min_{\alpha,\gamma,u,v} f_{L_M T_R KKM}(\alpha,\gamma,u,v) = \begin{bmatrix} \frac{\lambda}{2}\|\alpha\|^2 + \frac{1}{2}\|Y(\overline{A}\overline{A}^T Y\alpha - \overline{E}\gamma) - e\|^2 + \\ \frac{1}{2}\left[\|B_u^T u + I_u \overline{A}^T Y\alpha\|^2 + \|B_v^T v - I_v \overline{A}^T Y\alpha\|^2\right] + \\ \frac{1}{2}\left[\|B_{bu}^T u + E_u\gamma + e_u\|^2 + \|B_{bv}^T v - E_v\gamma + e_v\|^2\right] \end{bmatrix} \tag{5.2.3}$$

In problem (5.2.3) the dual variables aren't weighted using the kernel matrix, just the L_2 norm of α is minimized, while in problem (4.2.3) the dual variables are weighted using the kernel implying a direct representation of *w* in dual space. The reason for minimizing the L_2 norm of α will become obvious at the end of the derivation.

Problem (5.2.3) is still a linear classification problem, because $\bar{A}\bar{A}^T$ is a linear kernel. But, replacing the linear kernel by a nonlinear kernel $K(\bar{A},\bar{A}^T)$, where $K(\bar{A},\bar{A}^T)$ is a $m\times m$ matrix, problem (5.2.3) then becomes:

$$\min_{\alpha,\gamma,u,v} f_{NL_MT_RKKM}(\alpha,\gamma,u,v)=\begin{bmatrix}\frac{\lambda}{2}\|\alpha\|^2+\frac{1}{2}\|Y(K(\bar{A},\bar{A}^T)Y\alpha-\bar{E}\gamma)-e\|^2+\\ \frac{1}{2}\left[\|B_u^Tu+I_u\bar{A}^TY\alpha\|^2+\|B_v^Tv-I_v\bar{A}^TY\alpha\|^2\right]+\\ \frac{1}{2}\left[\|B_{bu}^Tu+E_u\gamma+e_u\|^2+\|B_{bv}^Tv-E_v\gamma+e_v\|^2\right]\end{bmatrix} \tag{5.2.4}$$

Setting

$$K_{m\times m}=K(\bar{A},\bar{A}^T) \text{ and } \bar{K}_{m\times m}=YKY \tag{5.2.5}$$

Problem (5.2.4) is given as:

$$\min_{\alpha,\gamma,u,v} f_{NL_MT_RKKM}(\alpha,\gamma,u,v)=\begin{bmatrix}\frac{\lambda}{2}\|\alpha\|^2+\frac{1}{2}\|\bar{K}\alpha-Y\bar{E}\gamma-e\|^2+\\ \frac{1}{2}\left[\|B_u^Tu+I_u\bar{A}^TY\alpha\|^2+\|B_v^Tv-I_v\bar{A}^TY\alpha\|^2\right]+\\ \frac{1}{2}\left[\|B_{bu}^Tu+E_u\gamma+e_u\|^2+\|B_{bv}^Tv-E_v\gamma+e_v\|^2\right]\end{bmatrix} \tag{5.2.6}$$

Definition 5.2 *Let* $A\in R^{m\times n}$ *and* $B\in R^{n\times k}$. *The* ***kernel*** $K(A,B)$ *maps* $R^{m\times n}\times R^{n\times k}$ *into* $R^{m\times k}$.

Definition 5.2 strongly relies on Mercer's condition (Burges, 1998; Cristianini & Shawe-Taylor, 2000) for symmetric kernel matrices.

Problem (5.2.4) is called the Nonlinear Multi-classification Tikhonov Regularization Knowledge-Based Kernel Machine (NL_MT_RKKM). It is a modification of problem (5.1.1.13) to accommodate nonlinearly separable patterns with knowledge sets, and explicit solutions in dual space in terms of the given data can be derived.

The NL_MT_RKKM model is a convex unconstrained optimization problem which has a unique minimum point. To find its minimum point, the optimality conditions need to be written explicitly.

Expanding each term in problem (5.2.6):

$$\frac{\lambda}{2}\|\alpha\|^2 = \frac{\lambda}{2}\alpha^T\alpha \tag{5.2.7}$$

$$\begin{aligned}\frac{1}{2}\left\|\bar{K}\alpha - Y\bar{E}\gamma - e\right\|^2 &= \frac{1}{2}\left[\bar{K}\alpha - Y\bar{E}\gamma - e\right]^T\left[\bar{K}\alpha - Y\bar{E}\gamma - e\right] \\ &= \frac{1}{2}\left[\alpha^T\bar{K}\bar{K}\alpha - 2\alpha^T\bar{K}Y\bar{E}\gamma - 2e^T\bar{K}\alpha + 2\gamma^T\bar{E}^TYe + \gamma^T\bar{E}^TYY\bar{E}^T\gamma + e^Te\right]\end{aligned} \tag{5.2.8}$$

$$\begin{aligned}\frac{1}{2}\left\|B_u^Tu + I_u\bar{A}^TY\alpha\right\|^2 &= \frac{1}{2}\left[B_u^Tu + I_u\bar{A}^TY\alpha\right]^T\left[B_u^Tu + I_u\bar{A}^TY\alpha\right] \\ &= \frac{1}{2}\left[u^TB_uB_u^Tu + 2u^TB_uI_u\bar{A}^TY\alpha + \alpha^TY\bar{A}I_u^TI_u\bar{A}^TY\alpha\right]\end{aligned} \tag{5.2.9}$$

$$\begin{aligned}\frac{1}{2}\left\|B_v^Tv - I_u\bar{A}^TY\alpha\right\|^2 &= \frac{1}{2}\left[B_v^Tv + I_v\bar{A}^TY\alpha\right]^T\left[B_v^Tv + I_v\bar{A}^TY\alpha\right] \\ &= \frac{1}{2}\left[v^TB_vB_v^Tv - 2v^TB_vI_v\bar{A}^TY\alpha + \alpha^TY\bar{A}I_v^TI_v\bar{A}^TY\alpha\right]\end{aligned} \tag{5.2.10}$$

$$\begin{aligned}\frac{1}{2}\left\|B_{bu}^T + E_u\gamma + e_u\right\|^2 &= \frac{1}{2}\left[B_{bu}^Tu + E_u\gamma + e_u\right]^T\left[B_{bu}^Tu + E_u\gamma + e_u\right] \\ &= \frac{1}{2}\begin{bmatrix}u^TB_{bu}B_{bu}^Tu + 2u^TB_{bu}E_u\gamma + \\ 2u^TB_{bu}e_u + 2e_u{}^TE_u\gamma + \gamma^TE_u{}^TE_u\gamma + e_u{}^Te_u\end{bmatrix}\end{aligned} \tag{5.2.11}$$

$$\begin{aligned}\frac{1}{2}\left\|B_{bv}^Tv - E_v\gamma + e_v\right\|^2 &= \frac{1}{2}\left[B_{bv}^Tv - E_v\gamma + e_v\right]^T\left[B_{bv}^Tv - E_v\gamma + e_v\right] \\ &= \frac{1}{2}\begin{bmatrix}v^TB_{bv}B_{bv}^Tv - 2v^TB_{bv}E_v\gamma + \\ 2v^TB_{bv}e_v - 2e_v{}^TE_v\gamma + \gamma^TE_v{}^TE_v\gamma + e_v{}^Te_v\end{bmatrix}\end{aligned} \tag{5.2.12}$$

Applying definition 5.2 and setting

$$K_{u_{m\times m}} = K(\bar{A}I_u{}^T, I_u\bar{A}^T) \text{ and } K_{Bu_{m\times g}} = K(\bar{A}I_u{}^T, B_u^T) \tag{5.2.13}$$

$$K_{v_{m\times m}} = K(\bar{A}I_v{}^T, I_v\bar{A}^T) \text{ and } K_{Bv_{m\times d}} = K(\bar{A}I_v{}^T, B_v^T) \tag{5.2.14}$$

To obtain a solution, kernel matrices K_u and K_v need to satisfy Mercer's conditions, i.e. K_u, K_v are symmetric and are of full rank (positive definite matrix). Kernel matrices K_{Bu} and K_{Bv} are rectangular kernels which cannot be symmetric. Hence, the kernel function given in definition 5.2 is very general.

Substituting kernels (5.2.13) and (5.2.14) into the expansion of the objective function, the optimality conditions for $f_{NL_M T_R KKM}(\alpha, \gamma, u, v)$ are obtained by setting the partial derivatives of (5.2.6) equal to zero, with respect to α, u, v and γ:

$$\frac{df}{d\alpha} = \lambda\alpha + \bar{K}\bar{K}\alpha - \bar{K}Y\bar{E}\gamma - \bar{K}^T e + YK_{Bu}u + YK_u Y\alpha - YK_{Bv}v + YK_v Y\alpha = 0 \qquad (5.2.15)$$

$$\frac{df}{du} = B_u B_u^T u + K_{Bu}{}^T Y\alpha + B_{bu}B_{bu}^T u + B_{bu}E_u\gamma + B_{bu}e_u = 0 \qquad (5.2.16)$$

$$\frac{df}{dv} = B_v B_v^T v - K_{Bv}{}^T Y\alpha + B_{bv}B_{bv}^T v - B_{bv}E_v\gamma + B_{bv}e_v = 0 \qquad (5.2.17)$$

$$\frac{df}{d\gamma} = \begin{bmatrix} -\bar{E}^T Y\bar{K}\alpha + \bar{E}^T Ye + \bar{E}^T YY\bar{E}\gamma + E_u{}^T B_{bu}^T u + \\ E_u{}^T e_u + E_u{}^T E_u\gamma - E_v{}^T B_{bv}^T v - E_v{}^T e_v + E_v{}^T E_v\gamma \end{bmatrix} = 0 \qquad (5.2.18)$$

To provide explicit matrix expressions for α and γ, the optimality conditions need to be rearranged. From equation (5.2.16) and (5.2.17) the following expressions are obtained for variables u and v:

$$\begin{aligned} &u = -UK_{Bu}^T Y\alpha - UB_{bu}E_u\gamma - UB_{bu}e_u, \\ &\text{where } U_{g\times g} = \left(B_u B_u^T + B_{bu}B_{bu}^T\right)^{-1} \end{aligned} \qquad (5.2.19)$$

$$\begin{aligned} &v = VK_{Bv}^T Y\alpha + VB_{bv}E_v\gamma - VB_{bv}e_v, \\ &\text{where } V_{d\times d} = \left(B_v B_v^T + B_{bv}B_{bv}^T\right)^{-1} \end{aligned} \qquad (5.2.20)$$

Since estimates are provided for u and v, the location (bias) of each pairwise hyperplane, relative to the origin in dual space, can be estimated by substituting (5.2.19) and (5.2.20) in equation (5.2.18):

$$\begin{aligned} &\gamma = \bar{H}(D\alpha - t) \\ &\text{where } \gamma = [\gamma^{(ij)}, ..., \gamma^{(k-1)k}]^T, \text{ for } i < j \end{aligned} \tag{5.2.21}$$

$$\bar{H}_{\bar{n}\times\bar{n}} = \left[\bar{E}^T Y Y \bar{E} + E_u{}^T \left[E_u - B_{bu}^T U B_{bu} E_u \right] + E_v{}^T \left[E_v - B_{bv}^T V B_{bv} E_v \right] \right]^{-1} \tag{5.2.22}$$

$$\begin{aligned} D_{\bar{n}\times m} &= \left[\bar{E}^T Y \bar{K} + E_u{}^T B_{bu}^T U K_{Bu}^T Y + E_v{}^T B_{bv}^T V K_{Bv}^T Y \right] \\ t_{\bar{n}\times 1} &= \left[\bar{E}^T Y e + E_u{}^T \left[e_u - B_{bu}^T U B_{bu} e_u \right] - E_v{}^T \left[e_v - B_{bv}^T V B_{bv} e_v \right] \right] \end{aligned} \tag{5.2.23}$$

With explicit solutions to u, v and γ already determined under the assumption that U, V and $\bar{H}$ are invertible matrices, when substituted into equation (5.2.15), the solution to the dual vector α can be conveniently expressed as:

$$\begin{aligned} &\alpha = \bar{M}^{-1}\bar{z} \\ &\text{where } \alpha = [\alpha^{(ij)T}, ..., \alpha^{(k-1)kT}]^T, \text{ for } i < j \end{aligned} \tag{5.2.24}$$

Decision function is given as:

$$D(x) = sign\left[K(x^T, A^{(ij)T}) Y^{(ij)} \alpha^{(ij)} - \gamma^{(ij)} \right] = \begin{cases} +1, & \text{if point } (x) \text{ is in class } A^{(i)} \\ -1, & \text{if point } (x) \text{ is in class } A^{(j)} \end{cases} \tag{5.2.25}$$

To classify a new point x, the voting approach strategy (Santosa, et al., 2002; Hsu & Lin, 2002) is employed. For example, if sign [$K(x^T, A^{(ij)T}) Y^{(ij)} \alpha^{(ij)} - \gamma^{(ij)}$] says that x is in the i^{th} class, then the vote for the i^{th} class is increased by one. Otherwise, the j^{th} is increased by one. Then we predict x as being in the class with the largest vote. In the case that those two classes have identical votes, select the one with the smallest index.

The corresponding linear system to solution (5.2.24) is:

$$\bar{M}\alpha = \bar{z} \tag{5.2.26}$$

Where

$$\bar{M}_{m\times m} = \begin{bmatrix} \lambda I + \bar{K}\left[\bar{K} - Y\bar{E}\bar{H}D\right] + \\ Y\left[K_u Y - K_{Bu} U\left[{K_{Bu}}^T Y + B_{bu} E_u \bar{H} D\right]\right] + \\ Y\left[K_v Y - K_{Bv} V\left[{K_{Bv}}^T Y + B_{bv} E_v \bar{H} D\right]\right] \end{bmatrix} \tag{5.2.27}$$

$$\bar{z}_{m\times 1} = \begin{bmatrix} \bar{K}^T\left[e - Y\bar{E}\bar{H}t\right] + \\ YK_{Bu} U B_{bu}\left[e_u - E_u \bar{H} t\right] - \\ YK_{Bv} V B_{bv}\left[e_v + E_v \bar{H} t\right] \end{bmatrix} \tag{5.2.28}$$

Minimizing the L_2 norm of α guarantees a solution for a positive tradeoff constant. However, minimizing the direct representation of w in dual space does not guarantee a solution for a positive tradeoff constant. This is the reason why the L_2 norm of α is minimized. It is evident from matrix $\bar{M}$ that only the diagonals change with any change in the tradeoff constant, implying we can always get a diagonally dominant matrix $\bar{M}$ which can ensure a solution. However, if the first term (identity matrix) in $\bar{M}$ is replaced with a kernel matrix, then for any constant, all elements in matrix $\bar{M}$ also increase. Therefore, we cannot guarantee a diagonally dominant matrix $\bar{M}$, hence guarantee a solution when minimizing the square norm of the linear combination of w in dual space.

Solution (5.2.24) provides an estimate to the linear system of equations in (5.2.26). Such a linear system is easier to solve than the quadratic programming formulation. Methods for solving the system include matrix decomposition methods and iterative based methods. Its solution involves the inversion of a dimensional matrix of the magnitude $m \times m$, where $m = \sum_{i<j}^{k} (m_i + m_j)$. Below is an algorithm for NL_MT_RKKM.

Algorithm 5.2 Nonlinear Multi-classification Tikhonov Regularization Knowledge-based Kernel Machine ($k \geq 2$): Given the data set of R^n that is represented by a matrix $A^{(i)} \in R^{m_i \times n}$, where $i = 1,..,k$ (k classes), and knowledge sets $\{x \mid B^{(i)}x \leq b^{(i)}\}$ or $\{x \mid \overline{B}^{(i)}x = \overline{b}^{(i)}\}$ belonging to $A^{(i)}$ $\{x \mid B^{(j)}x \leq b^{(j)}\}$ or $\{x \mid \overline{B}^{(j)}x = \overline{b}^{(j)}\}$ belonging to $A^{(j)}$. Let $A^{(i)}$ be a $m_i \times n$ matrix whose rows are points in the i^{th} class. Let $A^{(j)}$ be a $m_j \times n$ matrix whose rows are points in the j^{th} class. Weights α and γ for nonlinear classifier (5.2.25) is as follows.

NL_MT_RKKM Algorithm:

Step 1: Choose an appropriate kernel function such as a polynomial kernel function (2.1.3.4) or a Gaussian rbf kernel (2.1.3.5).

Step 2: Compute $\bar{M}$ from (5.2.27) for a specific constant λ.

Step 3: Compute $\bar{z}$ from (5.2.28).

Step 4: Determine α from (5.2.24).

Step 5: Determine γ from (5.2.21).

Step 6: Classify a new point x by (5.2.25).

It is in step 4 that matrix methods or iterative methods are invoked to obtain the solution to α.

5.3 Reduced Nonlinear Kernel Multi-classification Tikhonov Regularization Knowledge-based Machine

A drawback of NL_MT_RKKM model (5.2.6) is the lack of adequate memory storage of the parameters and solution. Such problems arise when dealing with very large data sets. It is in matrices (5.2.27) & (5.2.28) that the problems lie, because these equations involve the

addition of several large scale matrices, making the problem impractical to solve or even unsolvable, i.e. lacking an efficient method and/or solution. This is due to the lack of memory capacity for storing the data matrix and other associated parameters used, or necessary, in attaining a solution. To generate nonlinear classifiers the whole training dataset is used as the constraints, and a small subset of the training set is used as the dual variables of the optimization problem. This can be achieved by applying definition 5.2 to create a rectangular $m \times \hat{m}$ kernel $K(\overline{A}, \hat{A}^T)$.

Such a kernel matrix reduces the size of the optimization problem to be solved and simplifies the description of the pairwise nonlinear knowledge based separating surfaces.

Similar to the RK_MT_RM model (4.3.4), to reduce the number of dual variables associated with the dual representation of w, $\overline{A}^T$ can be replaced by $\hat{A}^T$ as below:

$$w = \hat{A}^T \hat{Y} \hat{\alpha} \tag{5.3.1}$$

Where $\hat{A}$ is a subset matrix similar to $\overline{A}$ in (4.2.4), but based on a percentage of the training set, i.e. $\hat{A}$ is subset matrix containing $\hat{A}^{(i)} \in R^{\hat{m}_i \times n}$, $\hat{A}^{(j)} \in R^{\hat{m}_j \times n}$, $i < j$ whose input vectors belong to the i^{th} and j^{th} subset; $\hat{Y}$ is a diagonal matrix similar to Y in (4.2.4), but whose $Y^{(ij)}$ matrix diagonals are ± 1 for the classes i and j subsets respectively, and α is the dual variable.

Consider minimizing the L_2 norm of $\hat{\alpha}$, and replacing $\overline{A}^T$ by $\hat{A}^T$ then problem (5.2.6) can be expressed as:

$$\min_{\hat{\alpha},\gamma,u,v} f_{RK_M T_R KM}(\hat{\alpha},\gamma,u,v) = \begin{bmatrix} \frac{\lambda}{2}\|\hat{\alpha}\|^2 + \frac{1}{2}\left\|Y(K(\bar{A},\hat{A}^T)\hat{Y}\hat{\alpha} - \bar{E}\gamma) - e\right\|^2 + \\ \frac{1}{2}\left[\left\|B_u^T u + I_u \hat{A}^T \hat{Y}\hat{\alpha}\right\|^2 + \left\|B_v^T v - I_v \hat{A}^T \hat{Y}\hat{\alpha}\right\|^2\right] + \\ \frac{1}{2}\left[\left\|B_{bu}^T u + E_u\gamma + e_u\right\|^2 + \left\|B_{bv}^T v - E_v\gamma + e_v\right\|^2\right] \end{bmatrix} \tag{5.3.2}$$

Setting

$$K_{m\times\hat{m}} = K(\bar{A},\hat{A}^T) \text{ and } \hat{K}_{m\times\hat{m}} = YK\hat{Y} \tag{5.3.3}$$

where $\hat{K}$ is a rectangular matrix.

Problem (5.3.2) is given as:

$$\min_{\hat{\alpha},\gamma,u,v} f_{RK_M T_R KM}(\hat{\alpha},\gamma,u,v) = \begin{bmatrix} \frac{\lambda}{2}\|\hat{\alpha}\|^2 + \frac{1}{2}\left\|\hat{K}\hat{\alpha} - Y\bar{E}\gamma - e\right\|^2 + \\ \frac{1}{2}\left[\left\|B_u^T u + I_u \hat{A}^T \hat{Y}\hat{\alpha}\right\|^2 + \left\|B_v^T v - I_v \hat{A}^T \hat{Y}\hat{\alpha}\right\|^2\right] + \\ \frac{1}{2}\left[\left\|B_{bu}^T u + E_u\gamma + e_u\right\|^2 + \left\|B_{bv}^T v - E_v\gamma + e_v\right\|^2\right] \end{bmatrix} \tag{5.3.4}$$

The RK_MT_RKM model (5.3.4) is a variant of NL_MT_RKKM model problem (5.2.6) that provides a sparse solution to a problem of interest, and explicit solutions in dual space in terms of the given data that can be derived. Its optimization problem is a convex objective function, which has a unique minimum point. To find its minimum point, the optimality conditions need to be written explicitly.

Expanding each term in problem (5.3.4):

$$\frac{\lambda}{2}\|\hat{\alpha}\|^2 = \frac{\lambda}{2}\hat{\alpha}^T\hat{\alpha} \tag{5.3.5}$$

$$\begin{aligned} \frac{1}{2}\left\|\hat{K}\hat{\alpha} - Y\bar{E}\gamma - e\right\|^2 &= \frac{1}{2}\left[\hat{K}\hat{\alpha} - Y\bar{E}\gamma - e\right]^T\left[\hat{K}\hat{\alpha} - Y\bar{E}\gamma - e\right] \\ &= \frac{1}{2}\left[\hat{\alpha}^T\hat{K}^T\hat{K}\hat{\alpha} - 2\hat{\alpha}^T\hat{K}^T Y\bar{E}\gamma - 2e^T\hat{K}\hat{\alpha} + 2\gamma^T\bar{E}^T Y^T e + \gamma^T\bar{E}^T Y^T Y\bar{E}^T\gamma + e^T e\right] \end{aligned} \tag{5.3.6}$$

$$\frac{1}{2}\left\|B_u^T u + I_u \hat{A}^T \hat{Y}\hat{\alpha}\right\|^2 = \frac{1}{2}\left[B_u^T u + I_u \hat{A}^T \hat{Y}\hat{\alpha}\right]^T \left[B_u^T u + I_u \hat{A}^T \hat{Y}\hat{\alpha}\right]$$
$$= \frac{1}{2}\left[u^T B_u B_u^T u + 2u^T B_u I_u \hat{A}^T \hat{Y}\hat{\alpha} + \hat{\alpha}^T \hat{Y}^T \hat{A} I_u^T I_u \hat{A}^T \hat{Y}\hat{\alpha}\right] \quad (5.3.7)$$

$$\frac{1}{2}\left\|B_v^T v - I_u \hat{A}^T \hat{Y}\hat{\alpha}\right\|^2 = \frac{1}{2}\left[B_v^T v + I_v \hat{A}^T \hat{Y}\hat{\alpha}\right]^T \left[B_v^T v + I_v \hat{A}^T \hat{Y}\hat{\alpha}\right]$$
$$= \frac{1}{2}\left[v^T B_v B_v^T v - 2v^T B_v I_v \hat{A}^T \hat{Y}\hat{\alpha} + \hat{\alpha}^T \hat{Y}^T \hat{A} I_v^T I_v \hat{A}^T \hat{Y}\hat{\alpha}\right] \quad (5.3.8)$$

$$\frac{1}{2}\left\|B_{bu}^T u + E_u \gamma + e_u\right\|^2 = \frac{1}{2}\left[B_{bu}^T u + E_u \gamma + e_u\right]^T \left[B_{bu}^T u + E_u \gamma + e_u\right]$$
$$= \frac{1}{2}\begin{bmatrix} u^T B_{bu} B_{bu}^T u + 2u^T B_{bu} E_u \gamma + \\ 2u^T B_{bu} e_u + 2e_u{}^T E_u \gamma + \gamma^T E_u{}^T E_u \gamma + e_u{}^T e_u \end{bmatrix} \quad (5.3.9)$$

$$\frac{1}{2}\left\|B_{bv}^T v - E_v \gamma + e_v\right\|^2 = \frac{1}{2}\left[B_{bv}^T v - E_v \gamma + e_v\right]^T \left[B_{bv}^T v - E_v \gamma + e_v\right]$$
$$= \frac{1}{2}\begin{bmatrix} v^T B_{bv} B_{bv}^T v - 2v^T B_{bv} E_v \gamma + \\ 2v^T B_{bv} e_v - 2e_v{}^T E_v \gamma + \gamma^T E_v{}^T E_v \gamma + e_v{}^T e_v \end{bmatrix} \quad (5.3.10)$$

Applying definition 5.2 and setting

$$\hat{K}_{u_{\hat{m}\times\hat{m}}} = K(\hat{A} I_u{}^T, I_u \hat{A}^T) \text{ and } \hat{K}_{Bu_{\hat{m}\times g}} = K(\hat{A} I_u{}^T, B_u^T) \quad (5.3.11)$$

$$\hat{K}_{v_{\hat{m}\times\hat{m}}} = K(\hat{A} I_v{}^T, I_v \hat{A}^T) \text{ and } \hat{K}_{Bv_{\hat{m}\times d}} = K(\hat{A} I_v{}^T, B_v^T) \quad (5.3.12)$$

Kernel matrices $\hat{K}_u$ and $\hat{K}_v$ need to satisfy Mercer's conditions, i.e. $\hat{K}_u, \hat{K}_v$ are symmetrical and are of full rank (positive definite matrix). Matrices $\hat{K}_{Bu}$ and $\hat{K}_{Bv}$ are rectangular kernel matrices similar to kernel (5.3.3) and cannot be symmetric. Substituting kernels (5.3.11) and (5.3.12) into the expansion of the objective function. The optimality conditions for $f_{RK_M T_R KM}(\hat{\alpha}, \gamma, u, v)$ are obtained by setting the partial derivatives of (5.3.4) equal to zero, with respect to $\hat{\alpha}, u, v$ and γ:

$$\frac{df}{d\hat{\alpha}} = \lambda\hat{\alpha} + \hat{K}^T \hat{K}\hat{\alpha} - \hat{K}^T Y \bar{E}\gamma - \hat{K}^T e + \hat{Y}^T \hat{K}_{Bu} u + \hat{Y}^T \hat{K}_u \hat{Y}\hat{\alpha} - \hat{Y}^T \hat{K}_{Bv} v + \hat{Y}^T \hat{K}_v \hat{Y}\hat{\alpha} = 0 \quad (5.3.13)$$

$$\frac{df}{du} = B_u B_u^T u + \hat{K}_{Bu}{}^T \hat{Y} \hat{\alpha} + B_{bu} B_{bu}^T u + B_{bu} E_u \gamma + B_{bu} e_u = 0 \tag{5.3.14}$$

$$\frac{df}{dv} = B_v B_v^T v - \hat{K}_{Bv}{}^T \hat{Y} \hat{\alpha} + B_{bv} B_{bv}^T v - B_{bv} E_v \gamma + B_{bv} e_v = 0 \tag{5.3.15}$$

$$\frac{df}{d\gamma} = \begin{bmatrix} -\bar{E}^T Y^T \hat{K} \hat{\alpha} + \bar{E}^T Y^T e + \bar{E}^T Y^T Y \bar{E} \gamma + E_u{}^T B_{bu}^T u + \\ E_u{}^T e_u + E_u{}^T E_u \gamma - E_v{}^T B_{bv}^T v - E_v{}^T e_v + E_v{}^T E_v \gamma \end{bmatrix} = 0 \tag{5.3.16}$$

To provide explicit matrix expressions for $\hat{\alpha}$ and γ, the optimality conditions need to be rearranged. From equation (5.3.14) and (5.3.15) the following expressions are obtained for variables u and v:

$$\begin{aligned} &u = -U\hat{K}_{Bu}^T \hat{Y} \hat{\alpha} - UB_{bu} E_u \gamma - UB_{bu} e_u, \\ &\text{where } U_{g \times g} = \left(B_u B_u^T + B_{bu} B_{bu}^T \right)^{-1} \end{aligned} \tag{5.3.17}$$

$$\begin{aligned} &v = V\hat{K}_{Bv}^T \hat{Y} \hat{\alpha} + VB_{bv} E_v \gamma - VB_{bv} e_v, \\ &\text{where } V_{d \times d} = \left(B_v B_v^T + B_{bv} B_{bv}^T \right)^{-1} \end{aligned} \tag{5.3.18}$$

Since estimates are provided for *u* and *v*, the location (bias) of each pairwise hyperplane, relative to the origin in dual space, can be estimated by substituting (5.3.17) and (5.3.18) in equation (5.3.16):

$$\begin{aligned} &\gamma = \hat{H}(\hat{D}\hat{\alpha} - \hat{t}) \\ &\text{where } \gamma = [\gamma^{(ij)}, ..., \gamma^{(k-1)k}]^T, \text{ for } i < j \end{aligned} \tag{5.3.19}$$

$$\hat{H}_{\bar{n} \times \bar{n}} = \left[\bar{E}^T Y^T Y \bar{E} + E_u{}^T \left[E_u - B_{bu}^T U B_{bu} E_u \right] + E_v{}^T \left[E_v - B_{bv}^T V B_{bv} E_v \right] \right]^{-1} \tag{5.3.20}$$

$$\begin{aligned} \hat{D}_{\bar{n} \times \hat{m}} &= \left[\bar{E}^T Y^T \hat{K} + E_u{}^T B_{bu}^T U \hat{K}_{Bu}^T \hat{Y} + E_v{}^T B_{bv}^T V \hat{K}_{Bv}^T \hat{Y} \right] \\ \hat{t}_{\bar{n} \times 1} &= \left[\bar{E}^T Y^T e + E_u{}^T \left[e_u - B_{bu}^T U B_{bu} e_u \right] - E_v{}^T \left[e_v - B_{bv}^T V B_{bv} e_v \right] \right] \end{aligned} \tag{5.3.21}$$

With explicit solutions to u, v and γ already determined under the assumption that U, V and $\hat{H}$ are invertible matrices, when substituted into equation (5.3.13) the solution to the dual vector $\hat{\alpha}$ can conveniently be expressed as:

$$\begin{aligned} &\hat{\alpha} = \hat{M}^{-1}\hat{z} \\ &\text{where } \hat{\alpha} = [\hat{\alpha}^{(ij)T}, ..., \hat{\alpha}^{(k-1)kT}]^T, \text{ for } i < j \end{aligned} \tag{5.3.22}$$

Decision function is given as:

$$D(x) = sign\left[K(x^T, \hat{A}^{(ij)T})\hat{Y}^{(ij)}\hat{\alpha}^{(ij)} - \gamma^{(ij)} \right] = \begin{cases} +1, & \text{if point } (x) \text{ is in class } A^{(i)} \\ -1, & \text{if point } (x) \text{ is in class } A^{(j)} \end{cases} \tag{5.3.23}$$

To classify a new point x, the voting approach strategy (Santosa, et al., 2002; Hsu & Lin, 2002) is employed. For example, if sign [$K(x^T, \hat{A}^{(ij)T})\hat{Y}^{(ij)}\hat{\alpha}^{(ij)} - \gamma^{(ij)}$] says that x is in the i^{th} class, then the vote for the i^{th} class is increased by one. Otherwise, the j^{th} is increased by one. Then we predict x as being in the class with the largest vote. In the case that those two classes have identical votes, select the one with the smallest index.

The corresponding linear system to solution (5.3.22) is:

$$\hat{M}\hat{\alpha} = \hat{z} \tag{5.3.24}$$

Where

$$\hat{M}_{\hat{m}\times\hat{m}} = \begin{bmatrix} \lambda I + \hat{K}^T\left[\hat{K} - Y\bar{E}\hat{H}\hat{D}\right] + \\ \hat{Y}^T\left[\hat{K}_u\hat{Y} - \hat{K}_{Bu}U\left[\hat{K}_{Bu}{}^T\hat{Y} + B_{bu}E_u\hat{H}\hat{D}\right]\right] + \\ \hat{Y}^T\left[\hat{K}_v\hat{Y} - \hat{K}_{Bv}V\left[\hat{K}_{Bv}{}^T\hat{Y} + B_{bv}E_v\hat{H}\hat{D}\right]\right] \end{bmatrix} \tag{5.3.25}$$

$$\hat{z}_{\hat{m}\times 1} = \begin{bmatrix} \hat{K}^T\left[e - Y\bar{E}\hat{H}\hat{t}\right] + \\ \hat{Y}^T\hat{K}_{Bu}UB_{bu}\left[e_u - E_u\hat{H}\hat{t}\right] - \\ \hat{Y}^T\hat{K}_{Bv}VB_{bv}\left[e_v + E_v\hat{H}\hat{t}\right] \end{bmatrix} \tag{5.3.26}$$

Solution (5.3.22) provides an estimate to the linear system of equations in (5.3.24). Such a linear system is easier to solve than the quadratic programming formulation. Methods for solving the system include matrix decomposition methods and iterative based methods. Its solution involves the inversion of a reduced dimensional matrix of the magnitude $\hat{m} \times \hat{m}$, where $\hat{m} = \sum_{i<j}^{k} (\hat{m}_i + \hat{m}_j)$. Below is an algorithm for RK_MT_RKM.

Algorithm 5.3 Reduced Nonlinear Kernel Multi-classification Tikhonov Regularization Knowledge-based Machine ($k \geq 2$): Given a data set in R^n that is represented by a matrix $A^{(i)} \in R^{m_i \times n}$, where $i = 1,..,k$ (k classes), and knowledge sets $\{x \mid B^{(i)}x \leq b^{(i)}\}$ or $\{x \mid \overline{B}^{(i)}x = \overline{b}^{(i)}\}$ belonging to $A^{(i)}$ $\{x \mid B^{(j)}x \leq b^{(j)}\}$ or $\{x \mid \overline{B}^{(j)}x = \overline{b}^{(j)}\}$ belonging to $A^{(j)}$. Let $A^{(i)}$ be a $m_i \times n$ matrix whose rows are points in the i^{th} class. Let $A^{(j)}$ be a $m_j \times n$ matrix whose rows are points in the j^{th} class.

Weights $\hat{\alpha}$ and γ for nonlinear classifier (5.3.23) is as follows.

RK_MT_RKM Algorithm:

Step 1: Choose a random subset matrix $\hat{A}^{(i)} \in R^{\hat{m}_i \times n}$, $\hat{A}^{(j)} \in R^{\hat{m}_j \times n}$, $i < j$ of the original data matrix $A^{(i)} \in R^{m_i \times n}$, $A^{(j)} \in R^{m_j \times n}$, $i < j$.

Step 2: Choose an appropriate kernel function such as a polynomial kernel function (2.1.3.4) or a Gaussian rbf kernel (2.1.3.5). Then compute its rectangular kernel matrix.

Step 3: Compute $\hat{M}$ from (5.3.25) for a specific constant λ.

Step 4: Compute $\hat{z}$ from (5.3.26).

Step 5: Determine $\hat{\alpha}$ from (5.3.22).

Step 6: Determine γ from (5.3.19).

Step 7: Classify a new point x by (5.3.23).

Chapter 6

Application

Discrimination of sets or objects is a practical problem in various fields such as finance (Trafalis & Ince, 2000), science (Ding & Dubchak, 2001; Santosa et al., 2002; Lee et al., 2003), political science (Malyscheff & Trafalis, 2003), and engineering (Trafalis & Oladunni, 2004; Trafalis et al., 2005). Discrimination requires the construction of classification models that can assign a data point or instance into a well-defined class.

6.1 Data Description

Below are the various data sets of interest:

Admission Data for Graduate School of Business: The Admission data set (Johnson & Wichern, chap. 11, 2002) uses the undergraduate grade point average (GPA) and graduate management aptitude test (GMAT) scores to help determine which applicants should be admitted to the school's graduate program. There are 85 instances (points) and 2 attributes (features), 43 data points used as training samples and 42 data points used as test samples. The distribution of instances with respect to their class is as follows: 28 instances in class 1 (not admitted), 26 instances in class 2 (borderline), and 31 instances in class 3 (admitted). The random sample validation was employed to validate the proposed models. In the random sample validation, fifty percent of the data set will be drawn randomly and trained on the proposed model, while the other 50 % will be used as test samples.

Iris flower Data: The Iris data set (Johnson & Wichern, chap. 11, 2002) uses the sepal length, sepal width, petal length, and petal width to help discriminate between three species of iris flower. There are 150 instances (points) and 4 attributes (features), 75 data points used as training samples and 75 data points used as test samples. The distribution of instances with respect to their class is as follows: 50 instances for each of the three classes. Class 1 belongs to the iris setosa specie, class 2 is the iris versicolor specie, and class 3 is the iris virginica specie. The random sample validation was employed to validate the proposed models.

Wine Recognition Data: The Wine data set uses the chemical analysis of wine grown in the same region in Italy to help discriminate between three different cultivars. There are 178 instances (points) and 13 attributes (features), 90 data points used as training samples, 88 data points used as test samples. The distribution of instances with respect to their class is as follows: 59 instances belong to class 1, 71 instances belong to class 2 and 48 instances belong to class 3. The random sample validation was employed to validate the proposed models. This data set is obtained from the UCI Repository of Machine Learning Databases and Domain Theories (Murphy & Aha, 1994).

Laminar/Turbulent Fluid Flow Data: Fluid flow pattern is determined by the generalized Reynolds number (N_{re}) thanks to the experimental work of Osborne Reynolds who determined the parameters that influence fluid flow (Bared, 1990). If $N_{re} < 2100$, then the flow is defined as laminar flow and if $N_{re} > 2100$, then the flow is defined as turbulent flow. The value 2100 is the critical Reynolds number which corresponds to the transition

from laminar to turbulent flow. Our interest is in the simulation of experimental data and Reynolds number equation for flow pattern in the annulus opposite the drill collars (Bourgoyne et al., 1991). The Reynold's equation is based on the Non-Newtonian bingham plastic model. The fluid flow data uses flow rate, density, viscosity, borehole diameter, drill collar diameter and mud type to delineate the flow pattern (laminar, +1; turbulent, -1) of the model. There are 92 data-points and 5 attributes, 46 instances for laminar flow pattern and 46 instances turbulent flow pattern (Trafalis & Oladunni, 2004; Oladunni & Trafalis, 2005).

The attributes are as follows:

1. ρ, density of fluid, ibm/gal – continuous variable
2. q, flow rate, gal/min – continuous variable
3. $(d_2 + d_1)$, summation of borehole and drill collar (OD) diameter, in – continuous variable
4. μ_p, plastic viscosity, cp – continuous variable
5. Mud type, water based mud (1) oil based mud (2) – categorical variable

Wisconsin Breast Cancer Prognosis Data: Tests were performed on the Wisconsin breast cancer prognosis dataset WPBC using a 60-month cutoff for predicting recurrence or non-recurrence of the disease (Bradley & Mangasarian, 1998).

Two validation methods were used: the random sample validation and the *k*-fold cross validation. In the random sample validation, fifty percent of the data set will be drawn randomly and trained on the proposed models, while the other 50 % will be used as test samples. In the *k*-fold cross validation, *k* subsets of equal sample size are created, and

training is performed on *k*-1 subsets, with the test set being the subset that is left out of training. The approach is performed *k* times until all subset have been gone through a training and test phase, but at different times. The average error is reported.

There are 198 instances (points) and 32 attributes (features), using random sampling validation, 100 data points used as training samples, 98 data points used as test samples. The distribution of instances with respect to their class is as follows: 47 instances in class 1 (recursive patients), 151 instances in class 2 (non-recursive patients). For data source, please refer to Fung et al. (2002).

For direct comparison with accuracy obtained from the KBSVM, a smaller subset of the WPBC data set is trained. The subset data has 110 instances (points) and 32 attributes (features). The distribution of instances with respect to their class is as follows: 41 instances in class 1 (recursive patients), 69 instances in class 2 (non-recursive patients). For data source, please refer to Fung et al. (2002).

Tornado Detection Data: There are three categories of severe weather; tornado detection, hail greater than 1.9cm in diameter, and non-tornadic winds in excess of 25m/s. Tornadoes have a rare occurrence but inflict a significant amount of damage to an area or vicinity. The Mesocyclone Detection Algorithm (MDA) data set is of interest, and it is based on the outputs from the Weather Surveillance Radar 1998 Doppler (WSR-88D) collected prior to the formation of a pre-tornadic circulation in the atmosphere. If there is a tornado reported, a circulation detected on a particular volume scan of radar data can be associated with that report.

The National Severe Storms Laboratory (NSSL) database labels tornado ground truth value based on temporal and spatial proximity. If there is a tornado occurrence between the beginning and end of the volume scan, and the occurrence is within a reasonable distance of circulation detection, then the ground truth value is flagged. However, if a circulation falls inside the prediction "time window" of -20 to +6 minutes of the ground truth occurrence duration, then the ground truth value is also flagged. Hence, the critical initiative involving these timings is to determine whether a circulation will produce a tornado within the next 20 minutes, a suitable lead time for advanced severe weather warnings by the National Weather Service.

The rule based SVM classification model by Trafalis et al. (2004) has shown promise in the tornado detection application. In that study, they have used a multiplier within the interval of 0.9 – 1.15 (increments of 0.05) to control the threshold value, and the best error rate (0.1449) was achieved at multiplier 1.05, i.e. each threshold value of the rules is multiplied by 1.05. The success of the model highlights the merits of a hybrid system incorporating both data and domain knowledge. With the data and prior knowledge, a model based on optimization theory is used to develop the hybrid model.

The Tornado data set uses 23 attributes based on Doppler velocity data (Trafalis et al., 2003). There are 1562 observations (points) and 23 attributes (features), 782 data points used as training samples and 780 data points used as test samples. The distribution of instances with respect to their class is as follows: 781 instances in class 1 (tornado), 781 instances in class 2 (non-tornado). The random sample validation was employed to validate the proposed models. The list of attributes used for tornado and non tornado discrimination are shown in Table 8.

Table 8. Mesocyclone Detection Algorithm (MDA) list of attributes and units

No.	Attributes	No.	Attributes
1	Base (m)	13	Low-level gate-to-gate velocity difference (m/s)
2	Depth (m)	14	Maximum gate-to-gate velocity difference (m/s)
3	Strength rank	15	Height of maximum gate-to-gate velocity difference (m)
4	Low-level diameter (m)	16	Core base (m)
5	Maximum diameter (m)	17	Core depth (m)
6	Height of maximum diameter (m)	18	Age (min)
7	Low-level rotational velocity (m/s)	19	Strength index (MSI) weighted by average density of integrated layer
8	Maximum rotational velocity (m/s)	20	Strength index (MSIr) "rank"
9	Height of maximum rotational velocity (m/s)	21	Relative depth (%)
10	Low-level shear (m/s/km)	22	Low-level convergence (m/s)
11	Maximum shear (m/s/km)	23	Mid-level convergence (m/s)
12	Height of maximum shear (m)		

Vertical Two-Phase Flow Data set (2D classification): The 2D vertical flow data set uses a pair of flow rates (superficial gas and liquid velocity) belonging to one inch pipes with fluid properties at atmospheric conditions, to delineate three different flow regimes. There are 209 instances (points) and 2 attributes (features), 107 data points used as training samples, 102 data points used as test samples. The distribution of instances with respect to their class is as follows: 44 instances in class 1 (bubble flow), 102 instances in class 2 (intermittent flow), and 63 instances in class 3 (annular flow). The random sample validation was employed to validate the proposed models. For data sources, please refer to Trafalis et al. (2005).

Vertical Two-Phase Flow Data set (3D classification): The 3D vertical flow data set uses a pair of flow rates (superficial gas and liquid velocity) and varying pipe sizes with fluid properties at atmospheric conditions, to delineate three different flow regimes. There are 424 instances (points) and 3 attributes (features), 206 data points used as training samples, 218 data points used as test samples. The distribution of instances with respect to their class is as follows: 98 instances in class 1 (bubble flow), 217 instances in class 2 (intermittent flow), and 109 instances in class 3 (annular flow). The random sample validation was employed to validate the proposed models. For data sources, please refer to Trafalis et al. (2005).

Horizontal Two-Phase Flow Data set (2D classification): The 2D horizontal flow data set uses a pair of flow rates (superficial gas and liquid velocity) belonging to one inch pipes with fluid properties at atmospheric conditions, to delineate five different flow regimes. There are 1195 instances (points) and 2 attributes (features), 597 data points used as training samples and 598 data points used as test samples. The distribution of instances with respect to their class is as follows: 415 instances in class 1 (annular flow), 25 instances in class 2 (dispersed bubble flow), 407 instances in class 3 (intermittent flow), 110 instances in class 2 (stratified smooth flow), and 238 instances in class 5 (stratified wavy flow). The random sample validation was employed to validate the proposed models. For data sources, please refer to Trafalis et al. (2005).

Horizontal Two-Phase Flow Data set (3D classification): The 3D horizontal flow data set uses a pair of flow rates (superficial gas and liquid velocity) and varying pipe sizes with

fluid properties at atmospheric conditions, to delineate five different flow regimes. There are 2272 instances (points) and 3 attributes (features), 1131 data points used as training samples and 1141 data points used as test samples. The distribution of instances with respect to their class is as follows: 699 instances in class 1 (annular flow), 81 instances in class 2 (dispersed bubble flow), 745 instances in class 3 (intermittent flow), 250 instances in class 2 (stratified smooth flow), and 497 instances in class 5 (stratified wavy flow). The random sample validation was employed to validate the proposed models. For data sources, please refer to Trafalis et al. (2005).

Table 9 illustrates the combinations of attributes used and the dimensions of the datasets.

Table 9. Vertical and Horizontal Flow Model Data Combination

Attributes	**2D classification**	**3D classification**
Gas superficial velocity	Yes	Yes
Liquid superficial velocity	Yes	Yes
Pipe diameter	No	Yes
Dimension of vertical dataset	**209 rows by 2 columns**	**424 rows by 3 columns**
Dimension of horizontal dataset	**1195 rows by 2 columns**	**2272 rows by 3 columns**

6.2 Prior Knowledge Descriptions

Below are the various prior knowledge sets of interest:

Laminar/Turbulent Fluid Flow Prior Knowledge (Reynold's Number): In addition to the fluid flow data, the Reynold's equation for a Bingham plastic model is used as prior knowledge to develop knowledge based classification model. As additional constraints, the prior knowledge will represent the transition equations which delineates laminar from turbulent flows. Since the flow pattern data are scaled by taking the natural logarithm of

each instance, the prior knowledge needs to be scaled by also considering a natural logarithm transformation of the equations. Below is the equivalent logarithmic transformation for Reynold's equation:

$$N_{re} = \frac{757\rho\bar{v}(d_2 - d_1)}{\mu_p}, \text{If } N_{re} < 2100 \rightarrow \text{Laminar, else if } N_{re} > 2100 \rightarrow \text{Turbulent}$$

$$\text{where } \bar{v} = \frac{q}{2.448(d_2^{\,2} - d_1^{\,2})} \quad , \quad (6.2.1)$$

$$\Rightarrow \begin{cases} \ln(\rho) + \ln(q) - \ln(d_2 + d_1) - \ln(\mu_p) \le 1.9156 - \varepsilon \rightarrow \text{Laminar} \\ \ln(\rho) + \ln(q) - \ln(d_2 + d_1) - \ln(\mu_p) \ge 1.9156 + \varepsilon \rightarrow \text{Turbulent} \end{cases}$$

where ε is a deviation factor, a small fraction perturbing the critical Reynold's number.

Wisconsin Breast Cancer Prognosis Prior Knowledge: In addition to the WPBC data, prior knowledge for both recursive and non-recursive patients are included to develop knowledge based classification models. The prior knowledge used is based on prognosis rules used by doctors (Lee et al., 2003). The rules actually depend on just two features, the diameter (cm) of the excised tumor (T), and the number of metastasized axillary lymph (L) nodes. Both features refer to attribute 31 and 32 of the WPBC dataset respectively. The rules are (Fung et al., 2002):

$$(L \ge 5) \wedge (T \ge 4) \Rightarrow RECURSIVE \quad (6.2.2)$$

$$(L = 0) \wedge (T \le 1.9) \Rightarrow NON - RECURSIVE \quad (6.2.3)$$

Tornado Prior Knowledge: In addition to the 23 MDA attributes data set, there are specific threshold values (Trafalis et al., 2004) assigned to each attribute for both classes. Usually these threshold values are determined by experts in the domain of meteorology.

For each attribute, the corresponding probability distribution function for tornado and non-tornado case arising from the training data are considered. The selection of the threshold for each rule was based on eliminating misclassification by inspecting if the minimum for a non-tornado case had a value less than the minimum for a tornado case for a particular attribute. If such a condition holds, then a region unique to non-tornadoes is found. Likewise, if the maximum for a non-tornado case had a value less than the maximum for a tornado case, for a particular attribute, then a region unique to tornado cases is found. Of 23 attributes, only 20 were found to be useful for rule generation (Trafalis et al., 2004). The thresholds used for tornado and non tornado discrimination are shown in Table 10.

Table 10. Threshold values for each MDA attribute. See (Trafalis et al., 2003) for description of attributes

Tornado thresholds	Non tornado thresholds
$x_1 < 90$	
$x_2 < 617$	$x_2 > 12219$
$x_3 < 0$	$x_3 > 13$
$x_4 < 813$	
$x_5 < 1091$	
$x_6 < 124$	
$x_7 < 6$	
$x_8 < 10$	
$x_9 < 122$	
$x_{10} < 2$	$x_{10} > 77$
	$x_{11} > 83$
$x_{12} < 106$	
$x_{13} < 3$	
$x_{14} < 11$	
$x_{15} < 122$	
$x_{16} < 106$	
$x_{17} < 617$	
	$x_{18} > 113$
	$x_{22} > 26$
	$x_{23} > 28$

Vertical and Horizontal Prior Knowledge (Flow Transitions): In addition to the vertical and horizontal flow data, prior knowledge for both flow models are included to develop knowledge based classification models. As additional constraints, the prior knowledge will represent the transition equations which delineate, or govern, the existence of each regime. Since the flow regime data are scaled by taking the natural logarithm of each instance, the prior knowledge needs to be scaled by also considering a natural logarithm transformation of the equations. Below are the transition equations (Trafalis et al., 2005) and their equivalent logarithmic transformations for 2D and 3D knowledge based classification.

Vertical flow correlations: The vertical two-phase flow transition equations used as expert knowledge for the knowledge based classification model are based on the work of McQuillan and Whalley (1985), and are as follows:

- Bubble – intermittent flow transition

$$\begin{aligned} & v_{LS} = 3.0 v_{GS} - 1.15 \left[\frac{g\sigma(\rho_L - \rho_G)}{\rho_L^2} \right]^{1/4} \\ & \Rightarrow \begin{cases} -0.9883 \ln(v_{GS}) + \ln(v_{LS}) = 1.0608, \text{ 2D classification} \\ -0.9883 \ln(v_{GS}) + \ln(v_{LS}) = 1.0608, \text{ 3D classification} \end{cases} \end{aligned} \tag{6.2.4}$$

- Bubble – dispersed bubble flow transition

$$\begin{aligned} & v_{LS} \geq \frac{6.8}{\rho_L^{0.444}} \{g\sigma(\rho_L - \rho_G)\}^{0.278} \left(\frac{D}{\mu_L} \right)^{0.112} \\ & \Rightarrow \begin{cases} \ln(v_{LS}) \geq 1.03345, \text{ 2D classification} \\ \ln(v_{LS}) \geq 1.4466 + 0.112 \ln(D), \text{ 3D classification} \end{cases} \end{aligned} \tag{6.2.5}$$

Horizontal flow correlations: The horizontal two-phase flow transition equations used as expert knowledge for the knowledge based classification model are based on the work of Weisman et al. (1979), and are as follows:

- Stratified – intermittent transition

$$\begin{aligned} & \frac{v_{GS}}{(gD)^{1/2}} = 0.25 \left(\frac{v_{GS}}{v_{LS}} \right)^{1.1} \\ & \Rightarrow \begin{cases} 0.1 \ln(v_{GS}) - 1.1 \ln(v_{LS}) = 2.0895, \text{ 2D classification} \\ 0.1 \ln(v_{GS}) - 1.1 \ln(v_{LS}) + 0.5 \ln(D) = 0.2451, \text{ 3D classification} \end{cases} \end{aligned} \tag{6.2.6}$$

- Stratified wavy – stratified smooth transition

$$\begin{aligned} & \left(\frac{\sigma}{gD^2 \Delta\rho} \right)^{0.20} \left(\frac{DG_G}{\mu_G} \right)^{0.45} = 8 \left(\frac{v_{GS}}{v_{LS}} \right)^{0.16}, \text{ where } \Delta\rho = \rho_L - \rho_G \\ & \Rightarrow \begin{cases} 0.29 \ln(v_{GS}) + 0.16 \ln(v_{LS}) = -0.4030, \text{ 2D classification} \\ 0.29 \ln(v_{GS}) + 0.16 \ln(v_{LS}) + 0.05 \ln(D) = -0.5875, \text{ 3D classification} \end{cases} \end{aligned} \tag{6.2.7}$$

- Transition to annular flow

$$1.9\left(\frac{v_{GS}}{v_{LS}}\right)^{1/8} = Ku^{0.2}Fr^{0.18} = \left(\frac{v_{GS}\rho_G^{1/2}}{[g(\rho_L-\rho_G)\sigma]^{1/4}}\right)^{0.2}\left(\frac{v_{GS}^2}{gD}\right)^{0.18}$$
$$\Rightarrow \begin{cases} 0.435\ln(v_{GS}) + 0.125\ln(v_{LS}) = 0.6911, \text{ 2D classification} \\ 0.435\ln(v_{GS}) + 0.125\ln(v_{LS}) - 0.18\ln(D) = 1.3551, \text{ 3D classification} \end{cases} \quad (6.2.8)$$

The correlations were selected based on the uncertainty of the transition lines as evidenced by the misclassification errors of the MSVM of Trafalis et al. (2005). Also, as a result of the uncertainty, the threshold values of the vertical and horizontal correlations where deviated by a small value, ±ε (deviation factor). This is in fact necessary because the correlation equations identify points on its boundaries, transitional points which have no distinct flow regime. So a small deviation of the thresholds facilitates the formation of bounds (deviated thresholds) for each flow regime. For instance equation (6.2.4), 2D case will be represented by:

- Bubble – intermittent flow transition

$$-0.9883\ln(v_{GS}) + \ln(v_{LS}) = 1.0608, \text{ 2D classification}$$
$$\Rightarrow \begin{cases} -0.9883\ln(v_{GS}) + \ln(v_{LS}) \geq 1.0608(1+\varepsilon) \rightarrow \text{Bubble} \\ -0.9883\ln(v_{GS}) + \ln(v_{LS}) \leq 1.0608(1-\varepsilon) \rightarrow \text{Intermitent} \end{cases} \quad (6.2.9)$$

Above, D is the pipe diameter, g is the acceleration due to gravity, v_{GS}, v_{LS} are the gas and liquid superficial velocities, respectively, ρ_G, ρ_L are the gas and liquid densities, σ is the surface tension, and μ_G, μ_L are the gas and liquid viscosities, respectively.

6.3 Forecast Assessment Indices for Tornado Detection

Since it is commonplace to use forecast assessment indices (Stephenson, 2000; Wilks, 1995), the results will be evaluated based on quantities in cells outlined in a contingency or confusion matrix (see Table 11).

Table 11. Tornado Forecast Confusion Matrix

		Observed		
		Yes	No	Total
Forecast	Yes	Hits (a)	False alarm (b)	Forecast "Yes"
	No	Misses (c)	Correct negative (d)	Forecast "No"
Total		Observed "Yes"	Observed "No"	

The cells (a, b, c, d) in the confusion matrix are used to compute the various forecast assessment indices. The indices used are as follows:

Misclassification error (error) is defined as

$$\text{error} = \frac{b+c}{a+b+c+d}$$

The misclassification error measures the fraction of misclassified points in the total number of observations. Its range is 0 to 1, and a perfect score is 0, which corresponds to 100% accuracy.

Probability of Detection (POD) is defined as

$$\text{POD} = \frac{a}{a+c}$$

The POD measures the fraction of observed events that were forecast correctly. Its range is 0 to 1, and a perfect score is 1 (or 100%). The POD is sensitive to hits, and ignores the false alarms. So for rare events the POD might be a better measure. The measure can be improved by including more "yes" forecasts to increase the number of hits.

False Alarm Rate (FAR) is defined as

$$\text{FAR} = \frac{b}{a+b}$$

The FAR measures the fraction of "yes" forecasts in which events did not occur. Its range is 0 to 1, and a perfect score is 0. The FAR is sensitive to false alarms, and ignores the misses. The measure can be improved by including more "no" forecasts to reduce the number of false alarms.

Bias is defined as

$$\text{Bias} = \frac{a+b}{a+c}$$

The bias measures the ratio of the frequency of forecast events to the frequency of observed events. Its range is 0 to infinity, and a perfect score is 1. The bias measures the tendency of the forecast system to over-forecast (bias > 1) or under-forecast (bias < 1) events. It does not measure how well the forecast corresponds to the observation, it measures only relative frequencies.

Heidke's Skill (HSS) is defined as

$$\text{HSS} = \frac{2(ad-bc)}{(a+b)(b+d)+(a+c)(c+d)}$$

The HSS is a measure of skill, where the concept of skill is one which a forecast is superior to some known reference forecast (e.g., random chance). Its range is -1 to +1, where -1 is an anti-skill, 0 is no skill, and +1 is a perfect skill. The HSS is commonly

used in meteorology since it uses all elements in the confusion matrix and works well for rare event forecasting (e.g. tornadoes) (Doswell et. al., 1990).

Probability of False Detection (POFD) is defined as

$$\text{POFD} = \frac{b}{b+d}$$

The POFD measures the ratio of false alarms to the total number of observations. Its range is 0 to 1, and a perfect score is 0. The POFD is a measure of inaccuracy with respect to the observations and provides a measure of the extent to which the forecasts provide a false warning for the occurrence of an event.

Critical Success Index (CSI) is defined as

$$\text{CSI} = \frac{a}{a+b+c}$$

The CSI measures the fraction of observed and/or forecast events that were correctly forecast. Its range is 0 to 1, and a perfect score is 1. The CSI is sensitive to hits, penalizes both misses and false alarms. The measure does not discriminate between sources of forecast error, and it depends on the climatological frequency of events (worse scores for rarer events) since some hits can occur purely due to random chance.

Chapter 7

Preliminary Computational Results

In this section, the results of the analyzed data sets described in chapter 6 are presented and discussed. The following model problems are used to train and test the data sets with and without prior knowledge.

- Linear Multi-classification Tikhonov Regularization Support Vector Machine (L_MT_RSVM): Model problem 4.1.9.
- Nonlinear Multi-classification Tikhonov Regularization Kernel Machine (NL_MT_RKM): Model problem 4.2.7.
- Reduced Nonlinear Kernel Multi-classification Tikhonov Regularization Machine (RK_MT_RM): Model problem 4.3.4.
- Linear Multi-classification Tikhonov Regularization Knowledge-based Support Vector Machine (L_MT_RKSVM): Model problem 5.1.1.13.
- Nonlinear Multi-classification Tikhonov Regularization Knowledge-based Kernel Machine (NL_MT_RKKM): Model problem 5.2.6.
- Reduced Nonlinear Kernel Multi-classification Tikhonov Regularization Knowledge-based Machine (RK_MT_RKM): Model problem 5.3.4.

All computations were performed using MATLAB. For nonlinear separation, a polynomial kernel function (2.1.3.4) with $p = 2$ was used to train and test the models. To demonstrate the uniqueness of the formulations, a comparison between the models was conducted. The comparisons were made by evaluating a performance parameter (misclassification error) defined below:

$$\beta = 1 - \left(\frac{Total\ number\ of\ correctly\ classified\ \ points}{Total\ number\ of\ observed\ \ points} \right) \tag{7.1}$$

β represents the overall misclassification error rate, i.e., the fraction of misclassified points for a given data set. For 100% classification, $\beta = 0$.

7.1 Data Driven Model

Results of the GPA, IRIS, WINE, 2D & 3D vertical, and horizontal flow data trained on the data driven models (L_MT_RSVM, NL_MT_RKM and RK_MT_RM), are shown in Tables 12 - 16. A comparison was also made on the multi-class LS-SVM model (3.2.7) (Suykens & Vandewalle, 1999c). It should be noted that β (error rate) in Tables 12 - 16 is defined by (7.1) and computing (cpu) time is measured in seconds.

Table 12. Average test error rate for L_MT_RSVM, NL_MT_RKM and multi-class LS-SVM on GPA Data (varying tradeoff, λ)

		Test Error Rate			
		GPA Data			
Statistics	**Tradeoff**	**L_MT_RSVM**	**LS-MSVM**	**NL_MT_RKM (poly kernel)**	**LS-MSVM (poly kernel)**
β	0 (0)	0.0952	-	-	-
Cpu time		*0.0067*	-	-	-
β	1 (0.01)	0.0952	0.0952	**0.0714**	0.0793
Cpu time		*0.0237*	*0.0167*	*0.0570*	*0.0433*
β	5 (0.05)	0.0793	**0.0714**	0.0873	0.0793
Cpu time		*0.0000*	*0.0133*	*0.0100*	*0.0200*
β	10 (0.1)	**0.0714**	**0.0714**	0.0873	0.0873
Cpu time		*0.0000*	*0.0233*	*0.0100*	*0.0233*
β	50 (0.5)	0.0873	**0.0714**	0.0793	0.0873
Cpu time		*0.0000*	*0.0503*	*0.0133*	*0.0167*
β	100 (1)	0.1032	0.0793	0.0793	**0.0714**
Cpu time		*0.0033*	*0.0100*	*0.0133*	*0.0100*

Table 13. Average test error rate for L_MT_RSVM, NL_MT_RKM and multi-class LS-SVM on IRIS Data (varying tradeoff, λ)

		Test Error			
		IRIS Data			
Statistics	Tradeoff	L_MT_RSVM	LS-MSVM	NL_MT_RKM (poly kernel)	LS-MSVM (poly kernel)
β	0 (0)	0.0355	-	-	-
Cpu time		*0.0033*	-	-	-
β	1 (0.01)	0.0267	0.0844	**0.0178**	0.0622
Cpu time		*0.0000*	*0.0540*	*0.0433*	*0.0833*
β	5 (0.05)	**0.0222**	**0.0400**	0.0311	0.0400
Cpu time		*0.0033*	*0.0233*	*0.0400*	*0.0433*
β	10 (0.1)	0.0355	**0.0400**	0.0356	0.0355
Cpu time		*0.0000*	*0.0200*	*0.0400*	*0.0467*
β	50 (0.5)	0.0711	0.0444	0.0444	**0.0222**
Cpu time		*0.0000*	*0.0200*	*0.0403*	*0.0633*
β	100 (1)	0.0978	0.0533	0.0444	0.0267
Cpu time		*0.0000*	*0.0233*	*0.0400*	*0.0507*

Table 14. Average test error rate for L_MT_RSVM, NL_MT_RKM and multi-class LS-SVM on WINE Data (varying tradeoff, λ)

		Test Error			
		WINE Data			
Statistics	Tradeoff	L_MT_RSVM	LS-MSVM	NL_MT_RKM (poly kernel)	LS-MSVM (poly kernel)
β	0 (0)	**0.0227**	-	-	-
Cpu time		*0.0067*	-	-	-
β	1 (0.01)	**0.0227**	0.0152	0.1174	**0.0303**
Cpu time		*0.0033*	*0.0870*	*0.0733*	*0.1370*
β	5 (0.05)	0.0303	**0.0114**	0.0644	0.0379
Cpu time		*0.0067*	*0.0667*	*0.0767*	*0.0667*
β	10 (0.1)	0.0303	**0.0114**	0.0341	0.0644
Cpu time		*0.0000*	*0.0733*	*0.0700*	*0.0667*
β	50 (0.5)	0.0379	0.0265	**0.0152**	0.0947
Cpu time		*0.0100*	*0.0400*	*0.0700*	*0.0667*
β	100 (1)	0.0341	0.0303	**0.0152**	0.1099
Cpu time		*0.0033*	*0.0433*	*0.0770*	*0.0867*

Table 15. Average test error rate for L_MT_RSVM, NL_MT_RKM, RK_MT_RM and multi-class LS-SVM on 2D & 3D VERTICAL Flow Data (varying tradeoff, λ)

		Test Error Rate									
		2D Vertical data					3D Vertical data				
Statistics	Tradeoff	L_MT_RSVM	LS-MSVM (linear kernel)	NL_MT_RKM (poly kernel)	LS-MSVM (poly kernel)	RK_MT_RM (poly kernel)	L_MT_RSVM	LS-MSVM (linear kernel)	NL_MT_RKM (poly kernel)	LS-MSVM (poly kernel)	RK_MT_RM (poly kernel)
β	0 (0)	**0.0098**	-	-	-	-	0.0493	-	-	-	-
Cpu time		*0.0067*	-	-	-	-	*0.0100*	-	-	-	-
β	1 (0.01)	**0.0098**	0.0163	**0.0131**	0.0163	**0.0131**	0.0493	**0.0425**	0.0321	0.0367	**0.0298**
Cpu time		*0.0000*	*0.1100*	*0.1133*	*0.1370*	*0.0300*	*0.0025*	*0.3880*	*0.7138*	*0.4203*	*0.2183*
β	5 (0.05)	**0.0098**	**0.0098**	**0.0131**	**0.0131**	**0.0131**	0.0493	0.0471	**0.0310**	**0.0310**	0.0310
Cpu time		*0.0000*	*0.0970*	*0.1070*	*0.1000*	*0.0333*	*0.0050*	*0.4208*	*0.6785*	*0.3803*	*0.2450*
β	10 (0.1)	**0.0098**	**0.0098**	**0.0131**	**0.0131**	**0.0131**	0.0493	0.0493	**0.0310**	**0.0310**	0.0310
Cpu time		*0.0033*	*0.0700*	*0.1133*	*0.1000*	*0.0300*	*0.0075*	*0.4305*	*0.6785*	*0.3908*	*0.1725*
β	50 (0.5)	0.0163	**0.0098**	0.0163	**0.0131**	**0.0131**	0.0459	0.0493	0.0367	**0.0310**	0.0356
Cpu time		*0.0300*	*0.0567*	*0.1200*	*0.1000*	*0.0303*	*0.0050*	*0.4455*	*0.6985*	*0.4178*	*0.2005*
β	100 (1)	0.0163	**0.0098**	0.0163	**0.0131**	**0.0131**	**0.0425**	0.0493	0.0367	0.0321	0.0356
Cpu time		*0.0000*	*0.0637*	*0.1003*	*0.0900*	*0.0733*	*0.0025*	*0.4260*	*0.6885*	*0.3810*	*0.2205*

Table 16. Average test error rate for L_MT_RSVM, NL_MT_RKM, RK_MT_RM and multi-class LS-SVM on 2D & 3D HORIZONTAL Flow Data (varying tradeoff, λ)

		Test Error Rate									
		2D Horizontal data					3D Horizontal data				
Statistics	Tradeoff	L_MT_RSVM	LS-MSVM (linear kernel)	NL_MT_RKM (poly kernel)	LS-MSVM (poly kernel)	RK_MT_RM (poly kernel)	L_MT_RSVM	LS-MSVM (linear kernel)	NL_MT_RKM (poly kernel)	LS-MSVM (poly kernel)	RK_MT_RM (poly kernel)
β	0 (0)	**0.0747**	-	-	-	-	0.1192	-	-	-	-
Cpu time		*0.0400*	-	-	-	-	*0.1003*	-	-	-	-
β	1 (0.01)	**0.0747**	0.1054	**0.0574**	0.0714	**0.0568**	0.1188	0.1266	**0.0763**	0.0850	**0.0736**
Cpu time		*0.0600*	*56.6213*	*37.5440*	*54.3047*	*12.8117*	*0.0903*	*243.9303*	*275.2483*	*248.6895*	*30.5895*
β	5 (0.05)	0.0758	0.0786	**0.0574**	0.0585	0.0574	0.1181	**0.1179**	0.0813	0.0842	0.0791
Cpu time		*0.0333*	*56.7247*	*37.6683*	*52.0647*	*12.6383*	*0.0953*	*244.5635*	*258.9593*	*247.1970*	*29.4490*
β	10 (0.1)	0.0775	0.0775	0.0591	0.0591	0.0574	**0.1179**	**0.1179**	0.0837	0.0837	0.0820
Cpu time		*0.0367*	*55.7867*	*38.3870*	*52.4323*	*12.8287*	*0.0903*	*244.3723*	*270.7075*	*249.2828*	*31.0585*
β	50 (0.5)	0.0892	**0.0747**	0.0608	**0.0574**	0.0569	0.1216	0.1188	0.0844	0.0778	0.0848
Cpu time		*0.0433*	*55.9607*	*37.7260*	*51.9917*	*12.6820*	*0.0900*	*245.6610*	*263.8185*	*248.0798*	*31.3438*
β	100 (1)	0.1054	**0.0747**	0.0714	**0.0574**	0.0586	0.1266	0.1188	0.0850	**0.0763**	0.0855
Cpu time		*0.0467*	*56.1707*	*37.8247*	*52.0150*	*12.6747*	*0.0953*	*247.5360*	*270.5538*	*247.2943*	*30.9410*

Observations

The values in bold are the lowest error rates for a specific data set with respect to its tradeoff constant. Tables 12 – 14 present the average test error rate results based on three runs (random samples) for the GPA, IRIS & WINE Data respectively. All models trained on the GPA data report similar error rates (0.0714, see Table 12). The use of the polynomial kernel function shows no improvement in the error rates. In Table 13, differences in the errors of the learning models become obvious. While all learning models report comparable error rates, the NL_MT_RKM model reports the lowest misclassification error (0.0178). Similar to the IRIS data, the WINE data (see Table 14) also reflects the differences in the error rates of the learning models. While the

misclassification errors are comparable, especially between the L_MT_RSVM, LS-MSVM and NL_MT_RKM models, Suykens & Vandewalle LS-MSVM (1999c) report a slightly lower error (0.0114) than the rest.

The computational training time (cpu time) of all learning models trained on the GPA, IRIS and WINE data are adequate, but with L_MT_RSVM reporting the lowest cpu time. The L_MT_RSVM model is of the magnitude of number of attributes *n* of a given dataset, while the NL_MT_RKM is of the magnitude of the number of data points *m*, hence the computational training time of the linear model (L_MT_RSVM and LS-MSVM with a linear kernel) ($m>>n$) will be lower than its nonlinear counterpart (LS-MSVM & NL_MT_RKM, both with a polynomial kernel function). However, if ($m<<n$) then the computational training time of the nonlinear kernel models will be lower than its linear counterpart. Changing the tradeoff constant λ may improve the generalization ability, however for the NL_MT_RKM model ($\lambda=0$), the matrix becomes nonsingular. Hence, its solution is not appropriate because it is not unique, i.e. there exist several alternate solutions to the same problem using the same method (NL_MT_RKM).

Table 15 presents the average test error rate results, based on three & four runs (random samples) for the 2D & 3D vertical flow data respectively. With respect to the 2D vertical flow data, the mean error rate is lowest at 0.0098. This error rate was obtained using both the L_MT_RSVM and LS-MSVM models. The second best error rates (0.0131) were obtained by performing training on the RK_MT_RM, NL_MT_RKM and LS-MSVM models. The highest error rate (0.0163) was reported by the NL_MT_RKM and LS-MSVM (poly kernel) models. The error rates for the 2D data are low enough to suggest that the data might be linearly separable. The theoretical correlation error rate for 2D vertical flow is

0.0163, which is comparable but not better than the learning models. The learning models perform better than the theoretical model.

The 3D vertical flow data classification is a more complex problem because an extra attribute accounting for the influences of the pipe size is included in the model. The error rates for the RK_MT_RM model appear to outperform the other models in comparison. The lowest error rate stands at 0.0298, which is also better than the 3D vertical flow theoretical model error rate (0.1227). It should be noted that the NL_MT_RKM and LS-MSVM formulations are in dual space which is induced by a kernel function, and the RK_MT_RM model formulation is in a reduced dual space also induced by a kernel function.

Table 16 presents the average test error rate results, based on three & four runs (random samples) for the 2D & 3D horizontal flow data respectively. For both 2D and 3D horizontal flow data classification, the best error rate (0.0568 and 0.0736 respectively) was obtained by performing training on the RK_MT_RM model. The other learning models also report comparable error rates but, preference is given to the model with the lowest error rate (RK_MT_RM). The 2D and 3D horizontal flow theoretical error rates are 0.0970 and 0.0918 respectively, while less than 10%, it does not outperform the nonlinear learning models.

The cpu time appears to be consistent irrespective of the varying tradeoff constants. From a computational point of view, the RK_MT_RM model is better for nonlinear separation, because it reduces the computation (cpu) time to train nonlinearly separable data. Based on the results given in Tables 15 & 16, the reduction in cpu time is within the interval 68 – 89%. The RK_MT_RM model is a significantly faster and simpler method to implement.

7.2 Prior Knowledge Model

7.2.1 Laminar/Turbulent Fluid Flow Prior Knowledge

The tradeoff constant considered was within the interval 0 – 100; and the deviation factor used was $\varepsilon = 0.01$. Further comparisons were made between the NL_MT_RKKM model, L_MT_RKSVM model, the traditional SVM (Vapnik, 1995; Trafalis & Oladunni, 2004), and the mixed integer programming kernel based classifiers ($MIPKC_{ps}$ & $MIPKC_p$, Oladunni & Trafalis, 2005).

Results of the fluid flow pattern data with prior knowledge information trained on the NL_MT_RKKM model, and compared with the L_MT_RKSVM, SVM, $MIPKC_{ps}$, & $MIPKC_p$ model (Oladunni & Trafalis, 2005), as shown in Tables 17 & 18. It should be noted that β (error rate) in the Tables are defined by (7.1), and the computing (cpu) time is measured in seconds. All computations and experiments were performed using MATLAB for NL_MT_RKKM, L_MT_RKSVM & SVM (Gunn, 1998) model and CPLEX OPL (ILOG, 2003) for the mixed integer programming kernel based models.

Table 17. Sample and Average random sample validation test error rate for NL_MT_RKKM on Fluid Flow Pattern Data (varying tradeoff, λ).

Statistics	Tradeoff	Test 1	Test 2	Test 3	Test 4	Average
Error	0	-	-	-	-	-
Cpu time		-	-	-	-	-
Error	1	0.0000	0.0278	0.0000	0.0000	0.0070
Cpu time		*0.0200*	*0.0100*	*0.0200*	*0.0100*	*0.0150*
Error	5	0.0000	0.0000	0.0000	0.0000	**0.0000**
Cpu time		*0.0400*	*0.0100*	*0.0100*	*0.0200*	*0.0200*
Error	10	0.0000	0.0000	0.0000	0.0000	**0.0000**
Cpu time		*0.0200*	*0.0800*	*0.0200*	*0.0100*	*0.0325*
Error	50	0.0000	0.0000	0.0000	0.0278	0.0070
Cpu time		*0.0100*	*0.0100*	*0.1310*	*0.0700*	*0.0553*
Error	100	0.0000	0.0000	0.0556	0.0278	0.0209
Cpu time		*0.0200*	*0.0100*	*0.0100*	*0.0200*	*0.0150*

Table 18. Average error, accuracy & cpu. time of NL_MT_RKKM, L_MT_RKSVM, MIPKC & SVM models.

Average	NL_MT_RKKM	L_MT_RKSVM	MIPKCps	MIPKCp	SVM
Error	0.0000	0.0139	0.0139	0.0348	0.0209
Accuracy	1.0000	0.9861	0.9861	0.9652	0.9792
Cpu time	0.0200	0.0100	49.2975	83.9400	0.4250

Observations

Tables 17 & 18 contain results for the NL_MT_RKKM, L_MT_RKSVM, $MIPKC_{ps}$, $MIPKC_p$, & SVM model on the fluid flow pattern classification data. The models in comparison generally report promising error rates. The NL_MT_RKKM model reports the smallest error (0), meaning that the testing data set was correctly classified (100% classification). The error rate (0.0139) for the model with prior knowledge (L_MT_RKSVM) performs in the same capacity as the data driven mixed integer programming model ($MIPKC_{ps}$). The computation time of both the NL_MT_RKKM and L_MT_RKSVM model are comparable, and both clearly outperform the other learning models. The SVM models report the next best computation time and the MIPKC model reports the worst time.

The fast solution of the L_MT_RKSVM model is due the formulation of the problem that is analyzed in the primal (input) space, i.e., the dimensionality of the classification problem is the number of attributes (variables) of the data set, in our case n = 5. The fast solution of the NL_MT_RKKM model is due to the small size of the test set (m = 46 test observations) and, when possible, the avoidance of iterative methods in computing its solution. The next best computation time to the NL_MT_RKKM model is the SVM model, and its fast solution is also due to the (m = 46 test observations), but in this case iterative methods are employed to find its solution. The MIPKC models require more time because their formulation is based on integer and mixed integer programming techniques

which generally perform more enumerations in obtaining a solution than their continuous counterparts such as SVM, L_MT_RKSVM and NL_MT_RKKM.

The influence of the tradeoff is on a slight decrease from $\lambda = 1$ to 5, at the tradeoff constant, $\lambda = 10$ to 100, the error rates increase slightly. At $\lambda = 0$, no error rates are reported because the solution is not unique, meaning the matrix of the system of equations is singular. But at $\lambda = 1$, we actually provide equal weighting to both the maximization of the margin between the laminar and turbulent flow patterns, and minimization of the empirical error of misclassification. As the tradeoff increases then the weighting between both the maximization of the margin and minimization of the training misclassification error becomes uneven. In this study, $\lambda = 5$ & 10 report the best misclassification error (100% correct classification).

7.2.2 Wisconsin Breast Cancer Prognosis Prior Knowledge

Results of the WPBC data with prior knowledge information trained on the L_MT_RKSVM model and compared with the L_MT_RSVM model (4.1.9) are shown in Tables 19-23. Comparisons were also made on the LS-SVM model (2.4.1) (Suykens & Vandewalle, 1999b) and the KBSVM model (2.6.2) (Fung, et al., 2002). The L_MT_RSVM, L_MT_RKSVM & LS-SVM computations were performed using MATLAB, while the KBSVM computation was conducted on Xpress-MP. It should be noted that the error rate in Tables 19-23 is defined by (7.1) and computing (cpu) time is measured in seconds.

Table 19. Sample and Average random sample validation test error rate for $L_M T_R$SVM on WPBC Data (varying tradeoff, λ)

Statistics	Tradeoff	Test 1	Test 2	Test 3	Test 4	Test 5	Average
Error	0	0.3061	0.2857	0.3061	0.3061	0.2857	0.2979
Cpu time		*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
Error	1	0.2551	0.2449	0.2449	0.2755	0.2551	0.2551
Cpu time		*0.0000*	*0.0000*	*0.0100*	*0.0000*	*0.0000*	*0.0020*
Error	5	0.2347	0.2449	0.2551	0.2857	0.2449	0.2531
Cpu time		*0.0000*	*0.0000*	*0.0000*	*0.0100*	*0.0000*	*0.0020*
Error	10	0.2041	0.2347	0.2449	0.2755	0.2347	0.2388
Cpu time		*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
Error	50	0.2245	0.2449	0.2143	0.2551	0.2245	0.2327
Cpu time		*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0000*
Error	100	0.2245	0.2245	0.2245	0.2347	0.2347	0.2286
Cpu time		*0.0000*	*0.0000*	*0.0000*	*0.0000*	*0.0100*	*0.0020*

Table 20. Sample and Average random sample validation test error rate for LS-SVM on WPBC Data (varying tradeoff, λ)

Statistics	Tradeoff	Test 1	Test 2	Test 3	Test 4	Test 5	Average
Error	0	-	-	-	-	-	-
Cpu time		-	-	-	-	-	-
Error	0.01	0.2449	0.2245	0.2347	0.2347	0.2653	0.2408
Cpu time		*0.0310*	*0.1600*	*0.1400*	*0.1800*	*0.0900*	*0.1202*
Error	0.05	0.2857	0.2143	0.2653	0.2449	0.2551	0.2531
Cpu time		*0.0100*	*0.0200*	*0.0600*	*0.0710*	*0.0200*	*0.0362*
Error	0.1	0.2959	0.2245	0.2449	0.2449	0.2653	0.2551
Cpu time		*0.0400*	*0.0800*	*0.0100*	*0.8000*	*0.0200*	*0.1900*
Error	0.5	0.2857	0.2551	0.2857	0.2653	0.2857	0.2755
Cpu time		*0.0300*	*0.0200*	*0.0100*	*0.0200*	*0.0500*	*0.0260*
Error	1	0.2857	0.2449	0.2857	0.2551	0.2857	0.2714
Cpu time		*0.0300*	*0.0200*	*0.1600*	*0.0400*	*0.0100*	*0.0520*

Table 21. Sample and Average random sample validation test error rate for $L_M T_R$KSVM on WPBC Data (varying tradeoff, λ)

Statistics	Tradeoff	Test 1	Test 2	Test 3	Test 4	Test 5	Average
Error	0	0.2653	0.2755	0.2449	0.2755	0.2347	0.2592
Cpu time		*0.0000*	*0.0600*	*0.0600*	*0.0500*	*0.0100*	*0.0360*
Error	1	0.2551	0.2755	0.2551	0.2755	0.2449	0.2612
Cpu time		*0.0100*	*0.0100*	*0.0100*	*0.0100*	*0.0100*	*0.0100*
Error	5	0.2245	0.2347	0.2551	0.2857	0.2347	0.2469
Cpu time		*0.0100*	*0.0000*	*0.0100*	*0.0100*	*0.0000*	*0.0060*
Error	10	0.2041	0.2347	0.2449	0.2755	0.2347	0.2388
Cpu time		*0.0100*	*0.0000*	*0.0100*	*0.0000*	*0.0100*	*0.0060*
Error	50	0.2347	0.2449	0.2143	0.2551	0.2143	0.2327
Cpu time		*0.0100*	*0.0100*	*0.0000*	*0.0100*	*0.0100*	*0.0080*
Error	100	0.2245	0.2245	0.2245	0.2449	0.2347	0.2306
Cpu time		*0.0100*	*0.0100*	*0.0100*	*0.0100*	*0.0100*	*0.0100*

Table 22. Sample and Average random sample validation test error rate for KBSVM on WPBC Data (varying tradeoff, λ)

Statistics	Tradeoff	Test 1	Test 2	Test 3	Test 4	Test 5	Average
Error	0	-	-	-	-	-	-
Cpu time		-	-	-	-	-	-
Error	1	0.2551	0.2959	0.2755	0.2959	0.2653	0.2775
Cpu time		*0.1000*	*0.1000*	*0.1000*	*0.1000*	*0.2000*	*0.1200*
Error	5	0.3980	0.3265	0.2245	0.2755	0.3265	0.3102
Cpu time		*0.1000*	*0.1000*	*0.1000*	*0.1000*	*0.1000*	*0.1000*
Error	10	0.3878	0.3776	0.2347	0.3367	0.3265	0.3327
Cpu time		*0.1000*	*0.1000*	*0.1000*	*0.1000*	*0.1000*	*0.1000*
Error	50	0.3776	0.3673	0.2653	0.3980	0.3265	0.3469
Cpu time		*0.1000*	*0.1000*	*0.1000*	*0.1000*	*0.1000*	*0.1000*
Error	100	0.3776	0.3673	0.3061	0.3980	0.3265	0.3551
Cpu time		*0.1000*	*0.1000*	*0.1000*	*0.1000*	*0.1000*	*0.1000*

Table 23. Average 10-fold cross validation test error rate for L_MT_RKSVM on WPBC Data set with 198 pts and 110 pts (varying tradeoff, λ)

		Average	
Statistics	**Tradeoff**	**$WPBC_{198}$**	**$WPBC_{110}$**
Error	0	0.2471	0.3091
Cpu time		*0.0882*	*0.0090*
Error	1	0.2521	0.3182
Cpu time		*0.0160*	*0.0080*
Error	5	0.2368	0.3182
Cpu time		*0.0150*	*0.0100*
Error	10	0.2268	0.3182
Cpu time		*0.0170*	*0.0070*
Error	50	0.2218	0.3182
Cpu time		*0.0160*	*0.0080*
Error	100	0.2168	0.3182
Cpu time		*0.0260*	*0.0060*

Observations

Tables 19 & 20 present the average test error rate results for the L_MT_RSVM & LS-SVM model respectively, based on five runs (random sample validation) for the WPBC (sample size = 198) without prior knowledge. The results show a decreasing trend of the error rates with an increase in the tradeoff constant for the L_MT_RSVM model. The L_MT_RSVM model reports a better error rate of 0.2286, while the LS-SVM lowest error rate is 0.2408.

Tables 21 & 22 present the average test error rate results for the L_MT_RKSVM & KBSVM models respectively, based on five runs (random sample validation) for the WPBC (sample size = 198) with prior knowledge. For the L_MT_RKSVM model, the results also show a decreasing trend of the error rates when the tradeoff constant increases. However, the KBSVM error rates show an increasing trend when the tradeoff constants increase.

The L_MT_RKSVM model reports a much better error rate of 0.2306, while the KBSVM lowest error rate is 0.2775.

For the tradeoffs ($\lambda > 0$) used in L_MT_RKSVM, the error rates do not appear to be significantly different. But at λ=0, a difference between the solution of the two models is observed. This suggests the prior rules do have a positive effect on the data. The stabilizer, which is the L_2 norm of w, is omitted (not minimized), and L_MT_RSVM only enforces closeness of data unique to its class (correctness of 70.21%), while the L_MT_RKSVM simultaneously enforces the closeness of data and prior knowledge unique to its class (correctness of 74.08%), and as a result the expert domain knowledge information is contained in the classifiers. The sample size of the data is small, so the computation running time is not an issue. Based on the random sampling validation results on Tables 19 - 22, the best and worst accuracy was reported by the L_MT_RSVM with correctness rate of 77.14% and 70.21% respectively. The correctness rate for the L_MT_RKSVM is 76.94% and 74.08% respectively. The length of the accuracy interval for both models is 6.93% for L_MT_RSVM, and 2.86% for the L_MT_RKSVM. If the interest lies in the highest accuracy, the best model would be L_MT_RSVM. However, since the highest accuracy for both models are approximately the same, the model with the lowest accuracy interval length shows more consistency, hence the L_MT_RKSVM is more favorable.

Table 23 presents the average test error rate results for the L_MT_RKSVM model, based on the 10-fold cross validation for the WPBC (sample size = 198, 110) with prior knowledge. The solution of the $WPBC_{198}$ data shows a decreasing trend of the error rates when the tradeoff constant increases. At λ = 100 (λ = 1), the highest and worst

correctness rate is 78.32% and 74.79% respectively, with an accuracy interval length of 3.53%. This analysis is consistent with random sample validation L_MT_RKSVM accuracy analysis in terms of the accuracy interval length, and clearly higher than the correctness rate of the L_MT_RSVM model. The solution of the $WPBC_{110}$ data shows an increasing trend of the error rates which approaches steady state when the tradeoff constant, $\lambda > 0$. At $\lambda = 0$ (λ = 1, 5, 10, 50 & 100), the highest (worst) correctness rate is 69.09% (68.18%) with accuracy interval length of 0.91%. The solution of the L_MT_RKSVM model reports a reasonably better solution of the Wisconsin breast cancer prognosis data set of 110 patients (recursive and non-recursive patient) than the KBSVM of Fung, et al. (2002), which has a correctness rate of 66.4%. It also outperforms the standard SVM model correctness rate of 66.2%.

The prior knowledge incorporation model (L_MT_RKSVM) can be considered a recipe for preventing over-fitting, because in addition to data sets, additional constraints representing a correlation or relationship inherent and unique to the existence of a category, is incorporated in the model. Hence, the results based on the L_MT_RKSVM model can be regarded as a more robust solution. The error rates of the *k*-fold cross validation method appear to be more consistent in terms of having both lower error rate and lower accuracy interval length.

7.2.3 Tornado Prior Knowledge

In this section, the forecast assessment indices defined in section 6.3 are used to analyze the tornado data sets and prior sets described in chapter 6. The L_MT_RKSVM model is used to train the data sets with prior knowledge. A comparison was also made on the

KBSVM model (2.6.2) (Fung, et al., 2002). The L_MT_RKSVM computation was performed using MATLAB, while the KBSVM computation was conducted on Xpress-MP. The average CPU time for the LT_RKSVM was 0.05s, whereas for the KBSVM method it was 0.6s Results from the testing data are summarized in Table 24 and 25.

Table 24. Forecast Summary for L_MT_RKSVM on Tornado Data

	L_MT_RKSVM					
Statistics	**Test 1**	**Test 2**	**Test 3**	**Test 4**	**Test 5**	**Average**
Error	0.1500	0.1474	0.1474	0.1397	0.1397	0.1449
POD	0.8795	0.8359	0.8615	0.8744	0.8692	0.8641
FAR	0.1695	0.1353	0.1537	0.1496	0.1461	0.1508
Bias	1.0590	0.9667	1.0179	1.0282	1.0179	1.0179
HSS	0.7000	0.7051	0.7051	0.7205	0.7205	0.7103
POFD	0.1795	0.1308	0.1564	0.1538	0.1487	0.1538
CSI	0.7457	0.7392	0.7450	0.7578	0.7567	0.7489

Table 25. Forecast Summary for KBSVM on Tornado Data

	KBSVM					
Statistics	**Test 1**	**Test 2**	**Test 3**	**Test 4**	**Test 5**	**Average**
Error	0.1744	0.1603	0.1667	0.1590	0.1769	0.1674
POD	0.8487	0.8282	0.8282	0.8462	0.8282	0.8359
FAR	0.1887	0.1522	0.1632	0.1624	0.1802	0.1694
Bias	1.0462	0.9769	0.9897	1.0103	1.0103	1.0067
HSS	0.6513	0.6795	0.6667	0.6821	0.6462	0.6651
POFD	0.1974	0.1487	0.1615	0.1641	0.1821	0.1708
CSI	0.7088	0.7210	0.7130	0.7269	0.7007	0.7141

Observations

Table 24 presents the sample and average summary statistics, based on five runs (random samples) for the Tornado Data with prior knowledge specified in section 6.2. The model data is a balanced design with equal data for training and testing, and the average values of all indices are favorable to the tornado data trained on the L_MT_RKSVM model.

The statistics of the L_MT_RKSVM model are very encouraging. The average error rate for five random samples is 0.1449, with a minimum error rate of 0.1397. The ability to forecast correctly observed events is represented by POD, which has an average value of

0.8641. Note that this is much closer to the ideal value of 1.0 than previous work (Marzban and Stumpf, 1996). Conversely, the ability to distinguish correct forecasts of tornadoes from those observations that do not materialize is summarized in the FAR. High FAR has been a persistent problem in tornado forecasting. Here we find that the average FAR is 0.1508, or about one-third of that reported in Marzban and Stumpf (1996). The relative contribution of POD and FAR can be assessed through the average bias, which has a value of 1.0179, indicating a slight over-forecast model; however, its value is close to 1, which is a perfect score. The incorrect forecasts can be interpreted further through POFD (0.1538), which examines observations of no tornadoes when forecasts were either for a tornado or no tornado. The perfect score for this index is 0. The CSI penalizes for both misses and false alarms. The average value is 0.7489. To determine how much better the forecast is than some random reference is measured by the skill. There is consistency in HSS values for all five random samples, the average HSS is 0.7103, and the perfect skill is 1.

These results can be compared directly to those for KBSVM model (see Table 25). The average error for KBSVM is 0.1674, a change that is approximately 15.5% higher than the error for the L_MT_RKSVM model. Similarly, the L_MT_RKSVM model outperforms the KBSVM with a higher POD (by +3.4%) and lower FAR (by -11%). Despite the slightly inferior performance of the KBSVM model, the bias is closer to the ideal 1.0, reflecting a slight reduction in over-forecasting at the expense of slightly increased under-forecasting. The L_MT_RKSVM has a notably lower POFD compared to KBSVM, with a change of -10%, whereas CSI drops similarly in KBSVM (-4.6%). Heidke Skill is markedly higher for L_MT_RKSVM (+6.8% change). Taken collectively, of these forecast evaluation

statistics suggest that $L_M T_R$KSVM is the most accurate technique. The results are promising, suggesting, the incorporation of prior knowledge to tornado discrimination offers a reasonably lower error rate, implying the thresholds values are good bounds for their respective attributes.

7.2.4 Vertical and Horizontal Prior Knowledge

The tradeoff constant considered are within the interval 0 – 100. The outputs (flow regimes) were coded as follows:

Vertical Flows: 1 – bubble, 2 – intermittent and 3 – annular. Slug and churn flows were grouped together as intermittent, because for both flows neither the gas nor the liquid phase is dominant.

Horizontal Flows: 1 – annular, 2 – dispersed bubble, 3 – intermittent, 4 – stratified smooth, and 5 – stratified wavy.

Results of the 2D & 3D vertical, and horizontal flow data with prior knowledge information trained on the data driven and knowledge based models, are shown in Tables 26-29. It should be noted that β (error rate) in the Tables is defined by (7.1), and the computing (cpu) time is measured in seconds. The theoretical correlations of McQuillan & Whalley (1985) and Weisman et al. (1979) were also simulated to compare with the learning models. The error rates for the 2D Vertical flow data is **0.0163**, the 3D Vertical flow data reports an error rate of **0.1227**, the 2D Horizontal flow data reports an error rate of **0.0970**, and the 3D Horizontal flow data reports an error rate of **0.0918**. All computations and experiments were performed using MATLAB.

Table 26. Average test error rate for RK_MT_RKM on 2D VERTICAL Flow Data (varying tradeoff, λ)

		Test Error Rate									
		2D VERTICAL Data									
Statistics	Tradeoff	L_MT_RSVM	L_MT_RKSVM _ε=0.01	NL_MT_RKM	NL_MT_RKKM _ε=0.01	RK_MT_RM	RK_MT_RKM_ ε=0.01	RK_MT_RKM_ ε=0.025	RK_MT_RKM_ ε=0.05	RK_MT_RKM_ ε=0.075	RK_MT_RKM_ ε=0.1
β	0	**0.0098**	**0.0098**	-	-	-	-	-	-	-	-
Cpu time		*0.0067*	*0.1000*	-	-	-	-	-	-	-	-
β	1	**0.0098**	**0.0098**	**0.0131**	**0.0131**	**0.0131**	**0.0131**	**0.0131**	**0.0131**	**0.0131**	0.0131
Cpu time		*0.0000*	*0.0267*	*0.1133*	*0.4063*	*0.0300*	*0.0540*	*0.0533*	*0.0333*	*0.0500*	*0.0803*
β	5	**0.0098**	**0.0098**	**0.0131**	**0.0131**	**0.0131**	**0.0131**	**0.0131**	**0.0131**	**0.0131**	0.0131
Cpu time		*0.0000*	*0.0067*	*0.1070*	*0.4013*	*0.0333*	*0.0767*	*0.0867*	*0.0533*	*0.0733*	*0.0300*
β	10	**0.0098**	**0.0098**	**0.0131**	**0.0131**	**0.0131**	**0.0131**	**0.0131**	**0.0131**	**0.0131**	0.0131
Cpu time		*0.0033*	*0.0067*	*0.1133*	*0.4113*	*0.0300*	*0.0500*	*0.1133*	*0.1200*	*0.0267*	*0.0400*
β	50	0.0163	0.0163	0.0163	**0.0131**	**0.0131**	**0.0131**	**0.0131**	0.0163	0.0229	**0.0098**
Cpu time		*0.0300*	*0.0100*	*0.1200*	*0.4010*	*0.0303*	*0.0400*	*0.0837*	*0.0670*	*0.1573*	*0.0700*
β	100	0.0163	0.0163	0.0163	**0.0131**	**0.0131**	**0.0131**	0.0163	**0.0131**	**0.0131**	0.0131
Cpu time		*0.0000*	*0.0400*	*0.1003*	*0.4063*	*0.0733*	*0.0933*	*0.1337*	*0.0500*	*0.0500*	*0.0800*

Table 27. Average test error rate for RK_MT_RKM on 3D VERTICAL Flow Data (varying tradeoff, λ)

		Test Error Rate									
		3D VERTICAL Data									
Statistics	Tradeoff	L_MT_RSVM	L_MT_RKSVM _ε=0.01	NL_MT_RKM	NL_MT_RKKM _ε=0.01	RK_MT_RM	RK_MT_RKM_ ε=0.01	RK_MT_RKM_ ε=0.025	RK_MT_RKM_ ε=0.05	RK_MT_RKM_ ε=0.075	RK_MT_RKM_ ε=0.1
β	0	0.0493	0.0493	-	-	-	-	-	-	-	-
Cpu time		*0.0100*	*0.1378*	-	-	-	-	-	-	-	-
β	1	0.0493	0.0493	0.0321	0.0356	**0.0298**	0.0356	0.0344	0.0459	0.0356	0.0424
Cpu time		*0.0025*	*0.0100*	*0.7138*	*2.6795*	*0.2183*	*0.1663*	*0.1503*	*0.1580*	*0.1500*	*0.1600*
β	5	0.0493	0.0493	**0.0310**	0.2259	0.0310	**0.0287**	0.0310	0.0298	**0.0298**	**0.0287**
Cpu time		*0.0050*	*0.0150*	*0.6785*	*2.6525*	*0.2450*	*0.1653*	*0.1578*	*0.1628*	*0.1530*	*0.1703*
β	10	0.0493	0.0482	**0.0310**	0.0470	0.0310	**0.0287**	**0.0298**	**0.0287**	**0.0298**	**0.0287**
Cpu time		*0.0075*	*0.0300*	*0.6785*	*2.6563*	*0.1725*	*0.1625*	*0.1575*	*0.1578*	*0.1528*	*0.1503*
β	50	0.0459	**0.0413**	0.0367	0.0333	0.0356	0.0333	0.0333	0.0333	0.0310	0.0333
Cpu time		*0.0050*	*0.0075*	*0.6985*	*2.7268*	*0.2005*	*0.1528*	*0.1578*	*0.1525*	*0.1378*	*0.1455*
β	100	**0.0425**	0.0425	0.0367	**0.0287**	0.0356	0.0333	0.0333	0.0321	0.0310	0.0344
Cpu time		*0.0025*	*0.0250*	*0.6885*	*2.7348*	*0.2205*	*0.1653*	*0.1678*	*0.1730*	*0.1625*	*0.1575*

Table 28. Average test error rate for RK_MT_RKM on 2D HORIZONTAL Flow Data (varying tradeoff, λ)

		Test Error Rate									
		2D HORIZONTAL Data									
Statistics	Tradeoff	L_MT_RSVM	L_MT_RKSVM _ε=0.01	NL_MT_RKM	NL_MT_RKKM _ε=0.01	RK_MT_RM	RK_MT_RKM_ ε=0.01	RK_MT_RKM_ ε=0.025	RK_MT_RKM_ ε=0.05	RK_MT_RKM_ ε=0.075	RK_MT_RKM_ ε=0.1
β	0	**0.0747**	**0.0758**	-	-	-	-	-	-	-	-
Cpu time		*0.0400*	*0.2137*	-	-	-	-	-	-	-	-
β	1	**0.0747**	**0.0758**	**0.0574**	**0.0541**	**0.0568**	**0.0541**	**0.0541**	**0.0541**	**0.0541**	**0.0541**
Cpu time		*0.0600*	*0.1237*	*37.5440*	*502.3743*	*12.8117*	*16.6740*	*16.9843*	*18.5330*	*17.2453*	*17.2913*
β	5	0.0758	0.0769	**0.0574**	**0.0541**	0.0574	**0.0541**	**0.0541**	**0.0541**	**0.0541**	0.0546
Cpu time		*0.0333*	*0.1000*	*37.6683*	*500.4393*	*12.6383*	*16.7507*	*17.0513*	*18.6467*	*17.4020*	*17.3013*
β	10	0.0775	0.0775	0.0591	**0.0541**	0.0574	**0.0541**	0.0546	0.0546	0.0546	0.0546
Cpu time		*0.0367*	*0.1033*	*38.3870*	*496.8027*	*12.8287*	*17.0877*	*17.1913*	*18.3497*	*17.4280*	*17.3413*
β	50	0.0892	0.0892	0.0608	**0.0541**	0.0569	0.0563	0.0574	0.0557	0.0557	0.0574
Cpu time		*0.0433*	*0.1100*	*37.7260*	*511.1840*	*12.6820*	*16.8007*	*16.9813*	*17.0010*	*17.3820*	*17.4953*
β	100	0.1054	0.1054	0.0714	0.0546	0.0586	0.0557	0.0596	0.0591	0.0574	0.0574
Cpu time		*0.0467*	*0.1267*	*37.8247*	*496.4187*	*12.6747*	*16.7873*	*17.2513*	*17.1243*	*17.4347*	*17.3017*

Table 29. Average test error rate for RK_MT_RKM on 3D HORIZONTAL Flow Data (varying tradeoff, λ)

		Test Error Rate									
		3D HORIZONTAL Data									
Statistics	Tradeoff	L_MT_RSVM	L_MT_RKSVM _ε=0.01	NL_MT_RKM	RK_MT_RM	RK_MT_RKM_ ε=0.01	RK_MT_RKM_ ε=0.025	RK_MT_RKM_ ε=0.05	RK_MT_RKM_ ε=0.075	RK_MT_RKM_ ε=0.1
β	0	0.1192	0.1177	-	-	-	-	-	-	-
Cpu time		*0.1003*	*0.4055*	-	-	-	-	-	-	-
β	1	0.1188	0.1177	**0.0763**	**0.0736**	**0.0793**	**0.0806**	**0.0789**	**0.0798**	**0.0795**
Cpu time		*0.0903*	*0.3230*	*275.2483*	*30.5895*	*56.7740*	*54.1403*	*54.1753*	*54.3215*	*54.1585*
β	5	0.1181	**0.1172**	0.0813	0.0791	0.0824	0.0828	0.0826	0.0828	0.0828
Cpu time		*0.0953*	*0.3180*	*258.9593*	*29.4490*	*56.4888*	*54.1988*	*54.1573*	*54.4073*	*54.1548*
β	10	**0.1179**	**0.1172**	0.0837	0.0820	0.0837	0.0830	0.0835	0.0835	0.0837
Cpu time		*0.0903*	*0.3180*	*270.7075*	*31.0585*	*56.6298*	*54.3508*	*54.2513*	*54.3253*	*54.4568*
β	50	0.1216	0.1205	0.0844	0.0848	0.0824	0.0824	0.0828	0.0824	0.0821
Cpu time		*0.0900*	*0.3080*	*263.8185*	*31.3438*	*56.7985*	*54.1750*	*54.1885*	*54.3448*	*54.1675*
β	100	0.1266	0.1267	0.0850	0.0855	0.1026	0.1606	0.1676	0.1976	0.1159
Cpu time		*0.0953*	*0.2588*	*270.5538*	*30.9410*	*56.9570*	*54.1905*	*54.2238*	*54.3565*	*55.3315*

Observations

The values in bold are the lowest error rates for each tradeoff constant λ.

Table 26 presents the average test error rate results, based on three runs (random samples) for the 2D vertical flow data. The lowest mean error rate (0.0098) was obtained using the L_MT_RSVM, L_MT_RKSVM and RK_MT_RKM model. The second best error rate (0.0131) was obtained by performing training on the NL_MT_RKM, NL_MT_RKKM and RK_MT_RM model. The error rates for the 2D data are low enough to suggest that the data might be linearly separable. The theoretical correlation error rate for 2D vertical flow is 0.0163, which is comparable but not better than the learning models. The 3D data classification is a more complex problem because an extra attribute accounting for the influences of the pipe size is included in the model.

Table 27 presents the average test error rate results, based on four runs (random samples) for the 3D vertical flow data. The error rate for the NL_MT_RKKM and RK_MT_RKM models appear to outperform the other models in comparison. The lowest error rate stands at 0.0287, which is also better that than the 3D vertical flow theoretical model error rate at 0.1227. It should be noted that the NL_MT_RKKM formulation are in a dual space induced by a kernel function, and the RK_MT_RKM model formulation is in a reduced dual space also induced by a kernel function.

Table 28 presents the average test error rate results, based on three runs (random samples) for the 2D horizontal flow data. The error rate for the NL_MT_RKKM and RK_MT_RKM models appear to outperform the other models in comparison. The lowest error rate stands at 0.0541 and it also surpasses the 3D vertical flow theoretical model error rate at 0.0970. The knowledge sets seem to influence the classifiers, because for

each nonlinearly data driven model, its corresponding knowledge based model reports a smaller error rate.

Table 29 presents the average test error rate results, based on four runs (random samples) for the 3D horizontal flow data. The lowest mean error rate (0.0736) was obtained by performing training on the RK_MT_RM model. The knowledge sets do not seem to influence the classifiers however, the reduction of the mapping input vectors of the model problems seem to influence the classifiers. The theoretical correlation error rate for 3D horizontal flow is 0.0918, which is comparable but not better than the learning models. The other learning models also reports comparable error rates but, preference is given to the model with the lowest error.

Overall the nonlinear models perform better than their linear counterparts. Incorporation of prior knowledge seems appropriate for the data sets. Through observation the nonlinear models with a higher cpu time perform better than the linear models. The cpu time of the NL_MT_RKKM is significantly higher than other methods. This is further testament that an increase in number of training samples, as well as the inclusion of additional constraints representing the knowledge set (prior knowledge), increases the computation (cpu) training time.

Models RK_MT_RM & RK_MT_RKM both use a reduced kernel matrix, and are both intended to decrease training time while providing simultaneously providing good generalization and reduced misclassification error. Reducing the dimensions of the kernel matrix reduces the computational time tremendously. The reduced kernel models are preferred due their fast implementation, ability to solve large data sets, and memory storage ability. Through experimentation, it is obvious that at $\lambda=0$ in dual space, the model is either

infeasible or nonsingular, and this applies to all nonlinear models. In either case their solutions are not acceptable, as it is does not exist or it is not unique.

The cpu time appear to be consistent irrespective of the varying tradeoff constants. From a computational point of view, the RK_MT_RKM model is better for nonlinear separation of sets with prior knowledge as it reduces the computation (cpu) time to train nonlinearly separable data. Based on the results given in Tables 26 - 29, the reduction in cpu time is within the interval 86 – 96%.

The prior knowledge incorporation model can be considered a recipe for preventing over-fitting, because in addition to data sets, additional constraints representing a correlation or relationship inherent and unique to the existence of a category, is incorporated in the model. Hence, the results based on the knowledge based models can be regarded as a more robust solution, capable of generating robust bounding planes for its category. Preference for a model should be based on the data set, i.e. having a visible structure or some similarity. For linear classification problems with (without) prior knowledge, model L_MT_RKSVM (L_MT_RSVM) is preferred. For nonlinear classification problems with (without) prior knowledge, model NL_MT_RKKM (NL_MT_RKM) is preferred. When computational time is essential to a nonlinear classification problem with (without) prior knowledge, model RK_MT_RKM (RK_MT_RM) is preferred.

Chapter 8

Summary, Conclusions and Future Research

8.1 Summary

In this study, several least square classification models have been developed for multi-category classification and knowledge based multi-category classification. The least square multi-classification models are referred to as the linear multi-classification Tikhonov regularization support vector machine (L_MT_RSVM), nonlinear multi-classification Tikhonov regularization kernel machine (NL_MT_RKM), reduced nonlinear kernel multi-classification Tikhonov regularization machine (RK_MT_RM), linear multi-classification Tikhonov regularization knowledge-based support vector machine (L_MT_RKSVM), nonlinear multi-classification Tikhonov regularization knowledge-based kernel machine (NL_MT_RKKM) and reduced nonlinear kernel multi-classification Tikhonov regularization knowledge-based machine (RK_MT_RKM). All six models are convex optimization problems, and they can be used to determine the maximum margin of separation between the given classes.

An important finding of the least square classification models is the ability to express the weights (parameters) that characterize the optimal separating hyperplane or surface, in terms of the given data of interest. Another positive outcome is the succinct representation of their respective formulations. The formulations express a multi-categorical problem with or without prior knowledge, as a single convex optimization. In addition, it can be used for binary problems or two-class classification problems with or without prior knowledge.

To illustrate the workings of the proposed models the binary AND problem, Exclusive XOR problem, and a simple three class problem with & without prior knowledge have been used. Through experimentation we showed the influence of the tradeoff parameter on the solution of the classification weights. An increase in the tradeoff increases the pairwise margin of separation defined by the approximate bounding hyperplanes and simultaneously reduces the L_2-norm of the normal vector defining the optimal hyperplane. More importantly, the influence of prior knowledge (knowledge sets) in the form of multiple polyhedral sets is demonstrated. In the present case, additional constraints that incorporate the prior knowledge set directly to the pairwise normal vectors have been formulated, and its solution is expressed in an explicit form.

The analysis of several data sets such as the GPA data, IRIS data, WINE data, WPBC data, Laminar/turbulent fluid flow pattern data, Tornado data, and Vertical & Horizontal Two-phase flow data look promising and in some cases outperform other existing models.

The ability of the models to provide fast solutions is very encouraging. In Tables 30 - 34 are the summary of the average error rates of all two-class and multi-class data sets (in bold are the lowest average error rates and cpu time for each data set). For the two-class model comparisons, all models yield comparable error rates. However, the knowledge-based models (L_MT_RKSVM & NL_MT_RKKM) report a better error rate in comparison with the rest of the models. For the multi-class model comparisons, the L_MT_RSVM & LS-MSVM yield comparable error rates. However, the reduced kernel data driven model and knowledge-based models (RK_MT_RM, L_MT_RKSVM, NL_MT_RKKM & RK_MT_RKM) report a better error rate in comparison with the rest of the models.

Table 30. Summary of average test error rate for SVM, L_MT_RKSVM & NL_MT_RKKM on two-class Flow pattern data

	Minimum Average Error Rate (cpu time)		
	Two-class Models		
Data	SVM	L_MT_RKSVM	NL_MT_RKKM
Flow pattern	0.0209 *(0.4250)*	0.0139 ***(0.01)***	**0** *(0.02)*

Table 31. Summary of average test error rate for L_MT_RSVM, LS-SVM, KBSVM & L_MT_RKSVM on two-class WPBC data (198 points)

	Minimum Average Error Rate (cpu time)			
	Two-class Models			
Data	L_MT_RSVM	LS-SVM	KBSVM	L_MT_RKSVM
WPBC	**0.2286** ***(0.002)***	0.2408 *(0.1202)*	0.2775 *(0.12)*	0.2306 *(0.01)*

Table 32. Summary of average test error rate for SVM, KBSVM & L_MT_RKSVM on two-class WPBC data (110 points)

	Minimum Average Error Rate (cpu time)		
	Two-class Models		
Data	SVM	KBSVM	L_MT_RKSVM
$WPBC_{110}$	0.3380	0.3360	**0.3091** ***(0.009)***

Table 33. Summary of average test error rate for L_MT_RSVM, SVM, KBSVM, & L_MT_RKSVM on two-class Tornado data

	Minimum Average Error Rate (cpu time)			
	Two-class Models			
Data	L_MT_RSVM	SVM	KBSVM	L_MT_RKSVM
Tornado	0.1611 ***(0.0033)***	0.1706	0.1674 *(0.6)*	**0.1449** *(0.05)*

Table 34. Summary of average test error rate for L_MT_RSVM, LS-MSVM, NL_MT_RKM, RK_MT_RM, L_MT_RKSVM, NL_MT_RKKM & RK_MT_RKM on multi-class data sets

	Minimum Average Error Rate (cpu time)							
	Multi-class Models							
	Data-driven Models					Knowledge-based Models		
Data	L_MT_RSVM	LS-MSVM (linear kernel)	NL_MT_RKM (poly kernel)	LS-MSVM (poly kernel)	RK_MT_RM (poly kernel)	L_MT_RKSVM	NL_MT_RKKM	RK_MT_RKM
GPA	0.0714 ***(0)***	0.0714 *(0.0133)*	0.0714 *(0.057)*	0.0714 *(0.01)*	**0.0635** ***(0)***	-	-	-
IRIS	0.0222 ***(0.0033)***	0.04 *(0.02)*	**0.0178** *(0.0433)*	0.0222 *(0.0633)*	0.0222 *(0.0208)*	-	-	-
WINE	0.0227 *(0.0033)*	**0.0114** *(0.0667)*	0.0152 *(0.07)*	0.0303 *(0.137)*	0.0227 ***(0)***	-	-	-
2D Vertical	**0.0098** ***(0)***	**0.0098** *(0.0567)*	0.0131 *(0.107)*	0.0131 *(0.1)*	0.0131 *(0.03)*	**0.0098** *(0.0067)*	0.0131 *(0.401)*	**0.0098** *(0.07)*
3D Vertical	0.0425 ***(0.0025)***	0.0425 *(0.338)*	0.0310 *(0.6785)*	0.0310 *(0.3803)*	0.0298 *(0.2183)*	0.0413 *(0.0075)*	**0.0287** *(2.7348)*	**0.0287** *(0.1503)*
2D Horizontal	0.0747 ***(0.0400)***	0.0747 *(55.9607)*	0.0574 *(37.544)*	0.0574 *(51.9917)*	0.0568 *(12.8117)*	0.0758 *(0.1237)*	**0.0541** *(496.4187)*	**0.0541** *(16.7507)*
3D Horizontal	0.1179 ***(0.0903)***	0.1179 *(244.3723)*	0.0763 *(275.2483)*	0.0763 *(247.2943)*	**0.0736** *(30.5895)*	0.1172 *(0.318)*	-	0.0789 *(54.1753)*

Note that the GPA, IRIS and WINE have no prior knowledge.

Overall, the knowledge-based models display a better performance in comparison with the other data driven (data-based) models. It is evident through experimentation that the L_MT_RSVM, RK_MT_RM, L_MT_RKSVM and RK_MT_RKM models are fast algorithms. The reduction in the computational time is a direct result of the derivation of the L_MT_RSVM & L_MT_RKSVM model problem to linear systems of equations of the magnitude of the attributes. However, the computational tractability of the RK_MT_RM and RK_MT_RKM is a direct result of the subset data used to induce the kernel function. It is also interesting to highlight the role of the tradeoff constant that can be used to force the models to be strong convex optimization problems, i.e., to satisfy positive definite matrix properties.

8.2 Future Research

Development of an algorithm for classification problem would be a good direction for future research. The linear system of equations derived from each optimization model problem can be investigated for determining a more efficient matrix method solution. Matrix methods such as LU decomposition, Cholesky decomposition, QR decomposition and singular value decomposition (SVD) (Lewis et al., 2006), or optimization algorithms such as conjugate gradient method (Lewis et al., 2006) can be investigated to solve this problem.

A recursive least square estimation can also be investigated in the context of online training/incremental learning methods. Such a model would avoid having to re-train the optimization model when a new point comes into the system/network. A recursive model would be computationally tractable since it can provide a means of storing the old solution, and then use the new point coming into the system to update the old solution.

Finally, since the proposed optimization models assume that the problem of interest is static/deterministic in nature, a dynamic/deterministic model can be investigated by assuming that each observation at time t_i is dependent on the observation at time t_{i-1}.

Bibliography

[1] Baesens B.; Viane, S.; Van Gestel, T.; Suykens, J. A. K.; Dedene, G.; De Moor, B.; Vanthiennen, J. An empirical assessment of kernel type performance for least squares support vector machine classifers, In *Proc. of the 4th International Conf. on Knowledge-Based Intelligent Engineering System and Allied Technologies (KES2000)*, Brighton, UK, Aug. **2000**.

[2] Bazaraa, M.S.; Sherali, H.D.; Shetty, C.M.; *Nonlinear Programming – Theory and Algorithms*. John Wiley & Sons, Inc., **1993.**

[3] Bennet, K.P.; Mangasarian O.L. Multicategory discrimination via linear programming, *Optimization Methods and Software* 3:27-39, **1994**.

[4] Bradley, P.S.; Mangasarian, O.L. Feature selection via concave minimization and support vector machines, Mathematical Programming Technical Report 98-03, February 1998. In J. Shavlik, editor, *Machine Learning Proceedings of the Fifteenth International Conference (ICML '98)*, Morgan Kaufmann, San Francisco, California, 82-90, **1998**.

[5] Bredensteiner E. J.; Bennet, K. P. Multicategory Classification by Support Vector Machines, *Computational Optimization and Applications* **1999**, 12, 53 – 79.

[6] Burges, C.J.C. A tutorial on support vector machines for pattern classification. *Data Mining and Knowledge Discovery,* 2(2): **1998**, 121-167.

[7] Chang, C-C.; Lin, C-J. *LIBSVM: A Library for Support Vector Machines* **2001**, http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[8] Cristianini, N.; Shawe-Taylor, J. *Support Vector Machines and other kernel-based learning methods*; Cambridge University Press, **2000**.

[9] Doswell, C.A. III.; Davies-Jones, R.; Keller, D. On summary measures of skill in rare event forecasting based on contingency tables. *Weather and Forecasting*, **1990**, 5, 576-585.

[10] Evgeniou, T.; Pontil, M.; Poggio, T. Regularization networks and support vector machines, *Advances in Computational Mathematics*, 13:1-50, **2000**.

[11] Fung, G.; Mangasarian, O.L. Proximal support vector machine classifiers. In D. Lee, F. Provost, and R. Srikant, editors, *Proceedings KDD2001: Knowledge Discovery and Data Mining*, August 26-29, **2001**, San Francisco, CA, pp. 64-70, New York, 2000. Asscociation for Computing Machinery. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-02.ps

[12] Fung, G.; Mangasarian, O.L.; Shavlik, J.W. Knowledge-Based support vector machine classifiers, Data Mining Institute Technical Report 01-09, November 2001. *Neural Information Processing Systems 2002 (NIPS 2002)*, Vancouver, BC, December 10-12, **2002**. ``Neural Information Processing Systems 15", S. Becker, S. Thrun and K. Obermayer, editors, MIT Press, Cambridge, MA, 2003, 521-528.

[13] Fung, G.; Mangasarian, O.L.; Shavlik, J.W. Knowledge-Based nonlinear kernel classifiers, Data Mining Institute Technical Report 03-02, March 2003. *Conference on Learning Theory (COLT 03) and Workshop on Kernel Machines*, Washington, D.C., August 24 - 27, 2003. Proceedings edited by Manfred Warmuth and Bernhard Scholkopf, Springer Verlag, Berlin, 102-113, **2003**.

[14] Fung, G.; Mangasarian, O.L. Multicategory proximal support vector machine classifiers, Data Mining Institute Technical Report 01-06, July 2001. *Machine Learning* 59:77-97, **2005**.

[15] Girosi, F. An equivalence between sparse approximation and support vector machines, *Neural Computation*, 10(6) 1455-1480, **1998.**

[16] Hsu, C-W.; Chang, C-C.; Lin, C-J. *A Practical Guide to Support Vector Classification*, Technical Report Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan, **2003**.

[17] Hsu, C-W.; Lin, C-J. A Comparison of Methods for Multi-class Support Vector Machines, *IEEE Transactions on Neural Networks,* **2002** 13, 415 – 425.

[18] Ivanov, V.V. *The Theory of Approximate Methods and Their Application to the Numerical Solution of Singular Integral Equations*. Nordhoff International, **1976**.

[19] Johnson, R. A.; Wichern, D. W. *Applied Multivariate Statistics Analysis*; Prentice Hall: New Jersey, **2002**.

[20] Lee, Y-J; Mangasarian, O.L.; Wolberg, W.H. Survival-time classification of breast cancer patients, Data Mining Institute Technical Report 01-03, March 2001. *Computational Optimization and Applications*, 25, **2003**, 151-166.

[21] Lewis, J. M.; Lakshmivarahan, S.; Dhall, S.; *Dynamic Data Assimilation*; Cambridge University Press, 2006.

[22] Mangasarian, O.L. *Nonlinear Programming*. SIAM, Philadelphia, PA, **1994**.

[23] Mangasarian, O.L. Generalized support vector machines, *Advances in Large Margin Classifiers*, (A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, eds.), MIT Press, Cambridge, MA, **2000**. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/99-14.ps.

[24] Mangasarian, O.L. Data mining via support vector machines, Data Mining Institute Technical Report 01-05, May **2001**. IFIP Conference on System Modelling and Optimization, Trier, Germany, July 23-27, 2001. System Modeling and Optimization XX,

E. W. Sachs and R. Tichatschke, editors, Kluwer Academic Publishers, Boston 2003, 91-112.

[25] Mangasarian, O.L. Knowledge-Based linear programming, Data Mining Institute Technical Report 03-04, July 2003. *SIAM Journal on Optimization*, 15, 375-382, **2005**.

[26] Mangasarian, O.L.; Shavlik, J.W.; Wild, E.W. Knowledge-Based kernel approximation, Data Mining Institute Technical Report 03-05, October 2003. *Journal of Machine Learning Research*, 5, 1127-1141, **2004**.

[27] *MATLAB User's Guide*. The Math-Works, Inc., Natwick, MA 01760, 1994-2003. http://www.mathworks.com.

[28] McQuillan, K.W.; Whalley, P.B. Flow Patterns in Vertical Two-Phase Flow, *Int. J. Multiphase Flow,* **1985**, 11, 161 – 175.

[29] Morozov, V.A. *Methods for Solving Incorrectly Posed Problems*. Springer-Verlag, **1984**.

[30] Murphy, P.M.; Aha, D.W. UCI repository of machine learning databases. [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Department of Information and Computer Science, University of California, Irvine, California, **1994**.

[31] Oladunni, O.; Trafalis, T.B. Least square multi-classification support vector machines: Pairwise (P_ALS-MSVM) & Piecewise (P_ILS-MSVM) formulations, *WSEAS Transactions on Circuits and Systems*, Issue 4**,** Vol. 4: 363-368, April**, 2005.**

[32] Pelckmans, K.; Suykens, J.A.K.; De Moor, B. Morozov, Ivanov and Tikhonov regularization based LS-SVMs, In *Proceedings of the International Conference On Neural Information Processing (ICONIP 2004)*, Calcutta, India, Nov. 22-25, **2004**.

[33] Platt, J.C.; Cristianini, N.; Shawe-Taylor, J. Largin margin DAGs for multiclass classification, In *Advances in Neural Information Processing Systems*, v. 12, pp. 547-553, MIT Press, **2000**.

[34] Reklaitis, G.V.; Ravindran, A.; Ragsdell, K.M. B.; *Engineering Optimization: Methods and Applications*. John Wiley & Sons, Inc., **1983.**

[35] Rifkin, R; Klautau, A. In Defense of One-Vs-All Classification, *Journal of Machine Learning Research,* Vol. 5, 101-141, **2004.**

[36] Santosa, B.; Conway, T.; Trafalis, T.B. Knowledge Based-Clustering and Application of Multi-Class SVM for Genes Expression Analysis, *Intelligent Engineering Systems through Artificial Neural Networks,* **2002**, 12, 391 – 395.

[37] Saunders, C.; Gammerman, A.; Vovk, V. Ridge regression learning algorithm in dual variables, *Proceedings of the 15th International Conference on Machine Learning IMCL-98,* Madison-Wisconsin, **1998**.

[38] Stephenson D.B. Use of "odds ratio" for diagnosing forecast skill. *Weather and Forecasting*, **2000**, 15, 221-232.

[39] Suykens, J. A. K.; Lukas, L.; Van Dooren, P.; De Moor, B.; Vandewalle, J. Least squares support vector machine classifers: A large scale algorithm, In *Proc. of the European Conf. on Circuit Theory and Design (ECCTD'99)*, 839-842, **1999.**

[40] Suykens, J.A.K.; Lukas, L.; Vandewalle, J. Sparse approximation using least squares support vector machines, In *Proc. of the IEEE International Symposium on Circuits and Systems (ISCAS'00)*, Geneva, Switzerland, pp. II 757-II 760, May, **2000.**

[41] Suykens, J.A.K.; Lukas, L.; Vandewalle, J. Sparse least squares support vector machine classifers, In *Proc. of the European Symposium on Artificial Neural Networks (ESANN'00)*, Bruges, Belgium, 37-42, **2000.**

[42] Suykens, J. A. K.; Vandewalle, J. Least Squares Support Vector Machine Classifers, *Neural Processing Letters*, 9, 293-300, **1999b.**

[43] Suykens, J. A. K.; Vandewalle, J. Multiclass least squares support vector machine classifers, In *Proc. of Joint Conf. on Neural Networks (IJCNN'99)*, Washington, D.C., **1999c.**

[44] Szedmak, S.; Shawe-Taylor, J.; Saunders, C.J.; Hardoon, D.R. Multiclass classification by L_1 norm support vector machine, In *Pattern Recognition and Machine Learning in Computer Vision Workshop*, Grenoble, France, May 02-04, **2004**.

[45] Tikhonov, A.N.; Arsenin, V.Y. Solution of Ill-Posed Problems. Winston, Washington D.C., **1977**.

[46] Trafalis, T.B.; Santosa, B.; Richman, M.B. Tornado Detection with Kernel-based Methods, *Intelligent Engineering Systems Through Artificial Neural Networks*, (C.H. Dagli, A.L. Buczak, J. Ghosh, M.J. Embrechts, and O. Ersoy, eds.), ASME Press, Vol. **13**: 677-682, **2003**.

[47] Trafalis, T.B.; Santosa, B.; Richman, M.B. Rule-Based Support Vector Machine Classifiers Applied to Tornado Prediction, *Computational Science-ICCS 2004*, Lecture notes in *Computer Science*, Springer, **2004**.

[48] Trafalis, T.B.; Oladunni, O. Pairwise Multi-classification Support Vector Machine: Quadratic Programming (QP-P_AMSVM) formulations, *WSEAS Transactions on Systems*, **4**:4, pp.349-354, April **2005**.

[49] Trafalis, T.B.; Oladunni, O.; Papavassiliou, D.V. Two-Phase Flow Regime Identification with a Multi-Classification SVM Model. *Industrial & Engineering Chemistry Research*, **2005**, 44, 4414 – 4426.

[50] Vapnik, V. *The Nature of Statistical Learning Theory*. New-York: Springer-Verlag, **1995**.

[51] Vapnik, V. *Statistical Learning Theory*. John Wiley & Sons, Inc., **1998**.

[52] Van Gestel, T.; Suykens, J. A. K.; Baesens B.; Viane, S.; Vanthiennen, J.; Dedene, G.; De Moor, B.; Vandewalle, J. Benchmarking least squares support vector machine classifers. *Machine Learning* **2004**, 54 (1), 5 – 32.

[53] Weisman, J.; Duncan, D.; Gibson, J.; Crawford, T. Effects of Fluid Properties and Pipe Diameter on Two-Phase Flow Patterns in Horizontal Lines, *Int. J. Multiphase Flow,* **1979**, 5, 437 – 462.

[54] Weston, J.; Watkins, C. Multi-class Support Vector Machines, In M. Verleysen, editor, *Proceedings of ESANN99*, Brussels, D. Facto Press, **1999**.

[55] Wilks, D.S. *Statistical Methods in the Atmospheric Sciences*. Academic Press, London, **1995**.

[56] Trafalis, T.B.; Ince, H. Support Vector Machine for Regression and Applications to Financial Forecasting, *IEEE-INNS-ENNS International Joint Conference on Neural Networks,* Como, Italy, July: 24-27, **2000**.

[57] Ding, C. H. Q.; Dubchak, I. Multi-class Protein Fold Recognition using support vector machines and neural networks. *Bioinformatics*, **2001**, 17, 349-358.

[58] Malyscheff, A. M.; Trafalis, T.B. Support Vector Machines and the Electoral College, *Proceedings of the International Joint Conference on Neural Networks*, Portland, Oregon, USA, IEEE Press, 2345-2348, **2003**.

[59] Trafalis, T.B.; Oladunni, O. Single Phase Fluid Flow Classification via Neural Networks & Support Vector Machine, *Intelligent Engineering Systems Through Artificial Neural Networks*, (C.H. Dagli, A.L. Buczak, D. L. Enke, M.J. Embrechts, and O. Ersoy, eds.), ASME Press, Vol. **14**: 427-432, **2004**.

[60] Bared, S.G. *The New Reynolds Number and the Estimated Drilling Rate under High Gravitations*, SPE 21630 (1990).

[61] Bourgoyne, A.T. Jr.; Chenevert, M.E.; Millheim, K.K.; Young, F.S. Jr. *Applied Drilling Engineering*, SPE Textbook Series Vol. 2, Richardson, TX, p. 137 – 144, (1991).

[62] Oladunni, O.; Trafalis, T.B. Mixed-Integer Programming Kernel-Based Classification, *WSEAS Transactions on Computers*, Issue **7**, Vol. **4**: 671-678, July, **2005**.

[63] Marzban C, Stumpf GJ (1996) A neural network for tornado prediction based on doppler radar-derived attributes. *Journal of Applied Meteorology*, 35:617-626

2003. http://www.mathworks.com.

[64] Oladunni, O.; Trafalis, T.B. Mixed-Integer Programming Kernel-Based Classification, *WSEAS Transactions on Computers*, Issue **7**, Vol. **4**: 671-678, July, **2005**.

[65] Gunn, S.T. (1998). *Support Vector Machine for Classification and Regression*, Technical Report, Department of Electronic and Computer Science, University of Southampton.

[66] ILOG OPL STUDIO 3.7. Language Manual, ILOG, S.A. Gentily, France and ILOG, Inc., Mountain View California, USA. **2003**.

http://www.ilog.com/products/oplstudio/

[67] Lee, Y-J; Mangasarian, O.L. RSVM: Reduced Support Vector Machines, *Proceedings of the First SIAM International Conference on Data Mining*, Chicago, Apr. 5-7, **2001**, CD-ROM Proceedings.

[68] Schölkopf; Smola A.J. *Learning with Kernels*, MIT Press, **2002**.

Appendices

Appendix A. L_MT_RSVM MATLAB Training Code (Linear Classification)

```
function [K,nbdata,gsol,xsol,t1]=Pair_wise_LMTRSVM(x,y,a)

% x: Input data (matrix)
% y: Class of input data (vector)
% a: trade off constant (scalar)
% usage: [K,nbdata,gsol,xsol,t1]=Pair_wise_LMTRSVM(x,y,a)

Mmax = max(y); %label: 1, 2, 3, ..., M
Mmin = min(y); %label: 1, 2, 3, ..., M
M = Mmax-Mmin + 1;
nbdata=zeros(1,M*(M-1)/2);
st = cputime;
[baris,col]=size(x);
xsup=[];
k=1;
K=[];Lt=[];
for i=1:M-1
    for j=i+1:M
        indi=find(y==i);
        indj=find(y==j);
        xapp=[x(indi,:); -x(indj,:)];
        xsup=[xsup;xapp];
        lab=[-ones(size(x(indi,:),1),1);ones(size(x(indj,:),1),1)];
        n2=size(xapp,1);
        nbdata(k)=n2;
        Lt=blkdiag(Lt,lab);
        K=blkdiag(K,xapp);
        k=k+1;
    end
end
sA=size(K);
[r,c]=size(Lt);
st=cputime;

% Linear Multiclass Tikhonov Regularization Support Vector Machine
Classification
xsol=(a*eye(sA(2))+(K'*K)-(K'*Lt*((Lt'*Lt)^-1)*(Lt'*K)))\((-
K'*Lt*((Lt'*Lt)^-1)*(Lt'*ones(sA(1),1)))+(K'*ones(sA(1),1))); % w
weights
gsol=((Lt'*Lt)^-1)*(-(Lt'*K*xsol)+(Lt'*ones(sA(1),1))); % bias offset
t1=cputime-st;
```

Appendix B. $L_M T_R$KSVM MATLAB Training Code (Linear Classification)

```
function [K,nbdata,gsol,xsol,t1]=Pair_wise_LMTRKSVM(x,y,B,b,yb,a)

% x: Input data (matrix)
% y: Class of input data (vector)
% B: Prior knowledge (matrix)
% b: Right hand side of prior knowledge (vector)
% yb: Class of prior knowledge (vector)
% a: trade off constant (scalar)
% usage: [K,nbdata,gsol,xsol,t1]=Pair_wise_LMTRKSVM(x,y,B,b,yb,a)

Mmax = max(y); %label: 1, 2, 3, ..., M
Mmin = min(y); %label: 1, 2, 3, ..., M
M = Mmax-Mmin + 1;
nbdata=zeros(1,M*(M-1)/2);
st = cputime;
[baris,col]=size(x);
xsup=[];
k=1;
K=[];Lt=[];Bti=[];Btj=[];IBi=[];IBj=[];bti=[];btj=[];Ibi=[];Ibj=[];ebi=
[];ebj=[];
for i=1:M-1
    for j=i+1:M
        indi=find(y==i);
        indj=find(y==j);
        indbi=find(yb==i);
        indbj=find(yb==j);
        KBi=B(indbi,:);
        KBj=B(indbj,:);
        Kbi=b(indbi,:);
        Kbj=b(indbj,:);
        laBi=eye(size(KBi,2));
        laBj=eye(size(KBj,2));
        labi=eye(size(Kbi,2));
        labj=eye(size(Kbj,2));
        xapp=[x(indi,:); -x(indj,:)];
        xsup=[xsup;xapp];
        lab=[-ones(size(x(indi,:),1),1);ones(size(x(indj,:),1),1)];
        n2=size(xapp,1);
        nbdata(k)=n2;
        Lt=blkdiag(Lt,lab);
        K=blkdiag(K,xapp);
        Bti=blkdiag(Bti,KBi); %Bi
        Btj=blkdiag(Btj,KBj); %Bj
        bti=blkdiag(bti,Kbi); %bi
        btj=blkdiag(btj,Kbj); %bj
        IBi=blkdiag(IBi,laBi); %Iu
        IBj=blkdiag(IBj,laBj); %Iv
        Ibi=blkdiag(Ibi,labi); %Eu
        Ibj=blkdiag(Ibj,labj); %Ev
        ebi=[ebi;labi];  %eu
        ebj=[ebj;labj];  %ev
        k=k+1;
    end
end
```

```
sA=size(K);
[r,c]=size(Lt);

% Linear Multiclass Tikhonov Regularization Knowledge Based Support
Vector Machine Classification
U=(Bti*Bti'+bti*bti')^-1;
V=(Btj*Btj'+btj*btj')^-1;
H=[(Lt'*Lt)+(Ibi'*(Ibi-bti'*U*bti*Ibi))+(Ibj'*(Ibj-btj'*V*btj*Ibj))]^-
1;
Li=(Lt'*K)-(Ibi'*bti'*U*Bti*IBi)-(Ibj'*btj'*V*Btj*IBj);
aa=(Lt'*ones(sA(1),1))-(Ibi'*(ebi-bti'*U*bti*ebi))+(Ibj'*(ebj-
btj'*V*btj*ebj));
M=[(a*eye(sA(2)))+(K'*(K-Lt*H*Li))+(IBi'*(IBi-Bti'*U*(Bti*IBi-
bti*Ibi*H*Li)))+(IBj'*(IBj-Btj'*V*(Btj*IBj-btj*Ibj*H*Li)))];
av=[(K'*(ones(sA(1),1)-Lt*H*aa))+(IBi'*Bti'*U*bti*(ebi+Ibi*H*aa))-
(IBj'*Btj'*V*btj*(ebj-Ibj*H*aa))];
xsol=M\av; % w weights
gsol=H*[-Li*xsol+aa]; % bias offset
t1=cputime-st;
```

Appendix C. L_MT_RSVM & L_MT_RKSVM MATLAB Testing Code (Linear Classification)

```
function yt=Pair_wise_LMTRSVM_test(K,tstx,nbdata,xsol,gsol)

%This to predict the label of data point with LMTRSVM or LMTRKSVM
% K: input data matrix obtained from Pairwise_LMTRSVM or
Pairwise_LMTRKSVM
% tstx: input sample for testing set
% nbdata: number of data for each classifier
% xsol: w weights obtained from Pairwise_LMTRSVM or Pairwise_LMTRKSVM
% gsol: bias offset obtained from Pairwise_LMTRSVM or Pairwise_LMTRKSVM
% yt: predicted label for testing set

[n1,n2]=size(tstx);
nbclass=(1+ sqrt(1+4*2*length(nbdata)))/2;
vote=zeros(n1,nbclass);
m=nbclass*(nbclass-1)/2;
k=1;
nbtest=[0 n1*ones(1,m)];
aux=cumsum(nbtest);
Knew=[];
for i=1:nbclass-1;
    for j=i+1:nbclass;
 Knew=blkdiag(Knew,tstx);
end
end
Knew;
size(Knew);
gamv=[];
for z=1:m
gam=[gsol(z)*ones(n1,1)];
gamv=[gamv;gam];
```

```
end
gamv;
size(gamv);
y=(Knew*xsol)-gamv;
for i=1:nbclass-1;
    for j=i+1:nbclass;
        indi=find(y(aux(k)+1:aux(k)+nbtest(k+1))>=0);
        indj=find(y(aux(k)+1:aux(k)+nbtest(k+1))<0);
        vote(indi,i)=vote(indi,i)+1;
        vote(indj,j)=vote(indj,j)+1;
        vote;
        k=k+1;
        if k==m
            break
        end
    end
end
[maxi,yt]=max(vote');
yt=yt';
```

Appendix D. NL_MT_RKM MATLAB Training Code (Nonlinear Classification)

```
function [K,Lt1,nbdata,gsol,xsol,t1]=Pair_wise_NLMTRKM(x,y,a,p,d)

% x: Input data (matrix)
% y: Class of input data (vector)
% a: trade off constant (scalar)
% p: polynomial kernel coefficent (scalar)
% d: degree of polynomial kernel (scalar)
% usage: [K,Lt1,nbdata,gsol,xsol,t1]=Pair_wise_NLMTRKM(x,y,a,p,d)

Mmax = max(y); %label: 1, 2, 3, ..., M
Mmin = min(y); %label: 1, 2, 3, ..., M
M = Mmax-Mmin + 1;
nbdata=zeros(1,M*(M-1)/2);
st = cputime;
[baris,col]=size(x)
xsup=[];
k=1;
K=[];Lt=[];Lt1=[];Y=[];
for i=1:M-1
    for j=i+1:M
        indi=find(y==i);
        indj=find(y==j);
        xapp=[x(indi,:); x(indj,:)];
        yapp=blkdiag(eye(length(indi)),-eye(length(indj)));
        xsup=[xsup;xapp];
        lab=[ones(size(x(indi,:),1),1);ones(size(x(indj,:),1),1)];
        lab1=[ones(size(x(indi,:),1),1);-ones(size(x(indj,:),1),1)];
        n2=size(xapp,1);
        nbdata(k)=n2;
        Lt=blkdiag(Lt,lab);
        Lt1=[Lt1;lab1];
        K=blkdiag(K,xapp);
```

```
        Y=blkdiag(Y,yapp);
        k=k+1;
    end
end
KKt=Y*((K*K'+p).^d)*Y;
E=Y*Lt;
sA=size(KKt);
[r,c]=size(Lt);

% Nonlinear Multiclass Tikhonov Regularization Kernel Machine
xsol=(a*eye(sA(2))+KKt-(E*((E'*E)^-1)*(E'*KKt)))\((-E*((E'*E)^-
1)*(E'*ones(sA(1),1)))+(ones(sA(1),1))); % alpha weights
gsol=((E'*E)^-1)*((E'*KKt*xsol)-(E'*ones(sA(1),1))); % bias offset
t1=cputime-st;
```

Appendix E. NL_MT_RKKM MATLAB Training Code (Nonlinear Classification)

```
function
[K,Lt1,nbdata,gsol,xsol,t1]=Pair_wise_NLMTRKKM(x,y,B,b,yb,a,p,d)

% x: Input data (matrix)
% y: Class of input data (vector)
% B: Prior knowledge (matrix)
% b: Right hand side of prior knowledge (vector)
% yb: Class of prior knowledge (vector)
% a: trade off constant (scalar)
% p: polynomial kernel coefficent (scalar)
% d: degree of polynomial kernel (scalar)
% usage:
[K,Lt1,nbdata,gsol,xsol,t1]=Pair_wise_NLMTRKKM(x,y,B,b,yb,a,p,d)

Mmax = max(y); %label: 1, 2, 3, ..., M
Mmin = min(y); %label: 1, 2, 3, ..., M
M = Mmax-Mmin + 1;
nbdata=zeros(1,M*(M-1)/2);
st = cputime;
[baris,col]=size(x);
xsup=[];
k=1;
K=[];Lt=[];Lt1=[];Y=[];
Bti=[];Btj=[];IBi=[];IBj=[];bti=[];btj=[];Ibi=[];Ibj=[];ebi=[];ebj=[];
for i=1:M-1
    for j=i+1:M
        indi=find(y==i);
        indj=find(y==j);
        indbi=find(yb==i);
        indbj=find(yb==j);
        KBi=B(indbi,:);
        KBj=B(indbj,:);
        Kbi=b(indbi,:);
        Kbj=b(indbj,:);
        laBi=eye(size(KBi,2));
        laBj=eye(size(KBj,2));
        labi=eye(size(Kbi,2));
```

```
        labj=eye(size(Kbj,2));
        xapp=[x(indi,:); x(indj,:)];
        yapp=blkdiag(eye(length(indi)),-eye(length(indj)));
        xsup=[xsup;xapp];
        lab=[ones(size(x(indi,:),1),1);ones(size(x(indj,:),1),1)];
        lab1=[ones(size(x(indi,:),1),1);-ones(size(x(indj,:),1),1)];
        n2=size(xapp,1);
        nbdata(k)=n2;
        Lt=blkdiag(Lt,lab);
        Lt1=[Lt1;lab1];
        K=blkdiag(K,xapp);
        Y=blkdiag(Y,yapp);
        Bti=blkdiag(Bti,KBi); %Bi
        Btj=blkdiag(Btj,KBj); %Bj
        bti=blkdiag(bti,Kbi); %bi
        btj=blkdiag(btj,Kbj); %bj
        IBi=blkdiag(IBi,laBi); %Iu
        IBj=blkdiag(IBj,laBj); %Iv
        Ibi=blkdiag(Ibi,labi); %Eu
        Ibj=blkdiag(Ibj,labj); %Ev
        ebi=[ebi;labi];  %eu
        ebj=[ebj;labj];  %ev
        k=k+1;
    end
end
KKt=Y*((K*K'+p).^d)*Y;
Ku=((K*IBi')*(IBi*K')+p).^d;
Kv=((K*IBj')*(IBj*K')+p).^d;
KBu=((K*IBi')*Bti'+p).^d;
KBv=((K*IBj')*Btj'+p).^d;
sA=size(KKt);
[r,c]=size(Lt);

% Nonlinear Multiclass Tikhonov Regularization Knowledge Based Kernel
Machine
U=inv(Bti*Bti'+bti*bti');
V=inv(Btj*Btj'+btj*btj');
Hbar=inv((Lt'*Y*Y*Lt)+(Ibi'*(Ibi-bti'*U*bti*Ibi))+(Ibj'*(Ibj-
btj'*V*btj*Ibj)));
D=(Lt'*Y*KKt)+(Ibi'*bti'*U*KBu'*Y)+(Ibj'*btj'*V*KBv'*Y);
t=(Lt'*Y*ones(sA(1),1))+(Ibi'*(ebi-bti'*U*bti*ebi))-(Ibj'*(ebj-
btj'*V*btj*ebj));
Mbar=[(a*eye(sA(2)))+(KKt*(KKt-Y*Lt*Hbar*D))+(Y*(Ku*Y-
KBu*U*(KBu'*Y+bti*Ibi*Hbar*D)))+(Y*(Kv*Y-
KBv*V*(KBv'*Y+btj*Ibj*Hbar*D)))];
zbar=[(KKt'*(ones(sA(1),1)-Y*Lt*Hbar*t))+(Y*KBu*U*bti*(ebi-
Ibi*Hbar*t))-(Y*KBv*V*btj*(ebj+Ibj*Hbar*t))];
xsol=Mbar\zbar; % alpha weights
gsol=Hbar*[D*xsol-t]; % bias offset
t1=cputime-st;
```

Appendix F. NL_MT_RKM & NL_MT_RKKM MATLAB Testing Code (Nonlinear Classification)

```
function yt=Pair_wise_NLMTRKM_test(K,Lt1,tstx,nbdata,xsol,gsol,p,d)

%This to predict the label of data point with NLMTRKM or NLMTRKKM
% K: input data matrix obtained from Pairwise_NLMTRKM or
Pairwise_NLMTRKKM
% tstx: input sample for testing set
% nbdata: number of data for each classifier
% xsol: alpha weights obtained from Pairwise_NLMTRKM or
Pairwise_NLMTRKKM
% gsol: bias offset obtained from Pairwise_NLMTRKM or Pairwise_NLMTRKKM
% p: polynomial kernel coefficent (scalar)
% d: degree of polynomial kernel (scalar)
% yt: predicted label for testing set

[n1,n2]=size(tstx);
nbclass=(1+ sqrt(1+4*2*length(nbdata)))/2;
vote=zeros(n1,nbclass);
m=nbclass*(nbclass-1)/2;
k=1;
nbtest=[0 n1*ones(1,m)];
aux=cumsum(nbtest);
Knew=[];
for i=1:nbclass-1;
    for j=i+1:nbclass;
 Knew=blkdiag(Knew,tstx);
end
end
Knew;
size(Knew);
gamv=[];
for z=1:m
gam=[gsol(z)*ones(n1,1)];
gamv=[gamv;gam];
end
gamv;
size(gamv);
H=(Knew*K'+p).^d;
size(H);
y=H*[xsol.*Lt1]-gamv;
for i=1:nbclass-1;
    for j=i+1:nbclass;
        indi=find(y(aux(k)+1:aux(k)+nbtest(k+1))>=0);
        indj=find(y(aux(k)+1:aux(k)+nbtest(k+1))<0);
        vote(indi,i)=vote(indi,i)+1;
        vote(indj,j)=vote(indj,j)+1;
        vote;
        k=k+1;
        if k==m
            break
        end
    end
```

```
end
[maxi,yt]=max(vote');
yt=yt';
```

Appendix G. RK_MT_RM MATLAB Training Code (Nonlinear Classification)

```
function [K1,Lt1,nbdata,gsol,xsol,t1]=Pair_wise_RKMTRM(x,y,x1,y1,a,p,d)

% x: Input data (matrix)
% y: Class of input data (vector)
% x1: A subset of input data (matrix)
% y1: A subset of class of input data (vector)
% a: trade off constant (scalar)
% p: polynomial kernel coefficent (scalar)
% d: degree of polynomial kernel (scalar)
% usage: [K1,Lt1,nbdata,gsol,xsol,t1]=Pair_wise_RKMTRM(x,y,x1,y1,a,p,d)

Mmax = max(y); %label: 1, 2, 3, ..., M
Mmin = min(y); %label: 1, 2, 3, ..., M
M = Mmax-Mmin + 1;
nbdata=zeros(1,M*(M-1)/2);
st = cputime;
[baris,col]=size(x)
xsup=[];
k=1;
K=[];K1=[];Lt=[];Ltx=[];Lt1=[];Y=[];Y1=[];
for i=1:M-1
    for j=i+1:M
        indi=find(y==i);
        indj=find(y==j);
        ind1i=find(y1==i);
        ind1j=find(y1==j);
        xapp=[x(indi,:); x(indj,:)];
        x1app=[x1(ind1i,:); x1(ind1j,:)];
        yapp=blkdiag(eye(length(indi)),-eye(length(indj)));
        y1app=blkdiag(eye(length(ind1i)),-eye(length(ind1j)));
        lab=[ones(size(x(indi,:),1),1);ones(size(x(indj,:),1),1)];
        lab1=[ones(size(x1(ind1i,:),1),1);-
ones(size(x1(ind1j,:),1),1)];
        n2=size(xapp,1);
        nbdata(k)=n2;
        Lt=blkdiag(Lt,lab);
        Lt1=[Lt1;lab1];
        K=blkdiag(K,xapp);
        K1=blkdiag(K1,x1app);
        Y=blkdiag(Y,yapp);
        Y1=blkdiag(Y1,y1app);
        k=k+1;
    end
end
KKt=Y*((K*K1'+p).^d)*Y1;
E=Y*Lt;
sA=size(KKt);
[r,c]=size(Lt);
```

```
% Reduced Kernel Multiclass Tikhonov Regularization Machine
xsol=(a*eye(sA(2))+(KKt'*KKt)-(KKt'*E*((E'*E)^-1)*(E'*KKt)))\((-
KKt'*E*((E'*E)^-1)*(E'*ones(sA(1),1)))+(KKt'*ones(sA(1),1))); % alpha
weights
gsol=((E'*E)^-1)*((E'*KKt*xsol)-(E'*ones(sA(1),1))); % bias offset
t1=cputime-st;
```

Appendix H. RK_MT_RKM MATLAB Training Code (Nonlinear Classification)

```
function
[K1,Lt1,nbdata,gsol,xsol,t1]=Pair_wise_RKMTRKM(x,y,x1,y1,B,b,yb,a,p,d)

% x: Input data (matrix)
% y: Class of input data (vector)
% x1: A subset of input data (matrix)
% y1: A subset of class of input data (vector)
% B: Prior knowledge (matrix)
% b: Right hand side of prior knowledge (vector)
% yb: Class of prior knowledge (vector)
% a: trade off constant (scalar)
% p: polynomial kernel coefficent (scalar)
% d: degree of polynomial kernel (scalar)
% usage:
[K1,Lt1,nbdata,gsol,xsol,t1]=Pair_wise_RKMTRKM(x,y,x1,y1,B,b,yb,a,p,d)

Mmax = max(y); %label: 1, 2, 3, ..., M
Mmin = min(y); %label: 1, 2, 3, ..., M
M = Mmax-Mmin + 1;
nbdata=zeros(1,M*(M-1)/2);
st = cputime;
[baris,col]=size(x);
xsup=[];
k=1;
K=[];K1=[];Lt=[];Ltx=[];Lt1=[];Y=[];Y1=[];
Bti=[];Btj=[];IBi=[];IBj=[];bti=[];btj=[];Ibi=[];Ibj=[];ebi=[];ebj=[];
for i=1:M-1
    for j=i+1:M
        indi=find(y==i);
        indj=find(y==j);
        ind1i=find(y1==i);
        ind1j=find(y1==j);
        indbi=find(yb==i);
        indbj=find(yb==j);
        KBi=B(indbi,:);
        KBj=B(indbj,:);
        Kbi=b(indbi,:);
        Kbj=b(indbj,:);
        laBi=eye(size(KBi,2));
        laBj=eye(size(KBj,2));
        labi=eye(size(Kbi,2));
        labj=eye(size(Kbj,2));
        xapp=[x(indi,:); x(indj,:)];
        x1app=[x1(ind1i,:); x1(ind1j,:)];
        yapp=blkdiag(eye(length(indi)),-eye(length(indj)));
```

```
        y1app=blkdiag(eye(length(ind1i)),-eye(length(ind1j)));
        lab=[ones(size(x(indi,:),1),1);ones(size(x(indj,:),1),1)];
        lab1=[ones(size(x1(ind1i,:),1),1);-
ones(size(x1(ind1j,:),1),1)];
        n2=size(xapp,1);
        nbdata(k)=n2;
        Lt=blkdiag(Lt,lab);
        Lt1=[Lt1;lab1];
        K=blkdiag(K,xapp);
        K1=blkdiag(K1,x1app);
        Y=blkdiag(Y,yapp);
        Y1=blkdiag(Y1,y1app);
        Bti=blkdiag(Bti,KBi); %Bi
        Btj=blkdiag(Btj,KBj); %Bj
        bti=blkdiag(bti,Kbi); %bi
        btj=blkdiag(btj,Kbj); %bj
        IBi=blkdiag(IBi,laBi); %Iu
        IBj=blkdiag(IBj,laBj); %Iv
        Ibi=blkdiag(Ibi,labi); %Eu
        Ibj=blkdiag(Ibj,labj); %Ev
        ebi=[ebi;labi];  %eu
        ebj=[ebj;labj];  %ev
        k=k+1;
    end
end
KKt=Y*((K*K1'+p).^d)*Y1;
Ku=((K1*IBi')*(IBi*K1')+p).^d;
Kv=((K1*IBj')*(IBj*K1')+p).^d;
KBu=((K1*IBi')*Bti'+p).^d;
KBv=((K1*IBj')*Btj'+p).^d;
sA=size(KKt);
[r,c]=size(Lt);
E=Y*Lt;

% Reduced Kernel Multiclass Tikhonov Regularization Knowledge Based
Machine
U=inv(Bti*Bti'+bti*bti');
V=inv(Btj*Btj'+btj*btj');
Hhat=inv((E'*E)+(Ibi'*(Ibi-bti'*U*bti*Ibi))+(Ibj'*(Ibj-
btj'*V*btj*Ibj)));
Dhat=(E'*KKt)+(Ibi'*bti'*U*KBu'*Y1)+(Ibj'*btj'*V*KBv'*Y1);
that=(E'*ones(sA(1),1))+(Ibi'*(ebi-bti'*U*bti*ebi))-(Ibj'*(ebj-
btj'*V*btj*ebj));
Mhat=[(a*eye(sA(2)))+(KKt'*(KKt-E*Hhat*Dhat))+(Y1'*(Ku*Y1-
KBu*U*(KBu'*Y1+bti*Ibi*Hhat*Dhat)))+(Y1'*(Kv*Y1-
KBv*V*(KBv'*Y1+btj*Ibj*Hhat*Dhat)))];
zhat=[(KKt'*(ones(sA(1),1)-E*Hhat*that))+(Y1'*KBu*U*bti*(ebi-
Ibi*Hhat*that))-(Y1'*KBv*V*btj*(ebj+Ibj*Hhat*that))];
xsol=Mhat\zhat; % alpha weights
gsol=Hhat*[Dhat*xsol-that]; % bias offset
t1=cputime-st;
```

Appendix I. RK_MT_RM & RK_MT_RKM MATLAB Testing Code (Nonlinear Classification)

```
function yt=Pair_wise_RKMTRM_test(K1,Lt1,tstx,nbdata,xsol,gsol,p,d)

%This to predict the label of data point with RKMTRM or RKMTRKM
% K1: input data matrix obtained from Pairwise_RKMTRM or
Pairwise_RKMTRKM
% tstx: input sample for testing set
% nbdata: number of data for each classifier
% xsol: alpha weights obtained from Pairwise_RKMTRM or Pairwise_RKMTRKM
% gsol: bias offset obtained from Pairwise_RKMTRM or Pairwise_RKMTRKM
% p: polynomial kernel coefficent (scalar)
% d: degree of polynomial kernel (scalar)
% yt: predicted label for testing set

[n1,n2]=size(tstx);
nbclass=(1+ sqrt(1+4*2*length(nbdata)))/2;
vote=zeros(n1,nbclass);
m=nbclass*(nbclass-1)/2;
k=1;
nbtest=[0 n1*ones(1,m)];
aux=cumsum(nbtest);
Knew=[];
for i=1:nbclass-1;
    for j=i+1:nbclass;
 Knew=blkdiag(Knew,tstx);
end
end
Knew;
size(Knew);

gamv=[];
for z=1:m
gam=[gsol(z)*ones(n1,1)];
gamv=[gamv;gam];
end
gamv;
size(gamv);

H=(Knew*K1'+p).^d;
size(H);
y=H*[xsol.*Lt1]-gamv;

for i=1:nbclass-1;
    for j=i+1:nbclass;
        indi=find(y(aux(k)+1:aux(k)+nbtest(k+1))>=0);
        indj=find(y(aux(k)+1:aux(k)+nbtest(k+1))<0);
        vote(indi,i)=vote(indi,i)+1;
        vote(indj,j)=vote(indj,j)+1;
        vote;
        k=k+1;
        if k==m
            break
```

```
        end
    end
end

[maxi,yt]=max(vote');
yt=yt';
```