UNIVERSITY OF CINCINNATI

DATE: AUGUST, 2003

I, RINAKO KAMEI, hereby submit this as part of the requirements for the degree of:

MASTER OF SCIENCE

in:

ELECTRICAL & COMPUTER ENGINEERING AND COMPUTER SCIENCE

It is entitled:

EXPERIMENTS IN PIECEWISE APPROXIMATION OF CLASS BOUNDARY USING SUPPORT VECTOR MACHINES

Approved by:

Dr. ANCA RALESCU
Dr. QING-AN ZENG
Dr. RAJ BHATNAGAR
Dr. DAN RALESCU
Experiments in Piecewise Approximation
of Class Boundary
using Support Vector Machines

A thesis submitted to the
Division of Research and Advanced Studies
of the University of Cincinnati
in partial fulfillment of the requirements
for the degree of
MASTER OF SCIENCE

in the Department of
Electrical and Computer Engineering
and Computer Science
of the College of Engineering
August 2003

by

Rinako Kamei

Bachelor of Informatics, March 2000,
Kansai University, Osaka, Japan.

Thesis Advisor & Committee Chair: Prof. Anca Ralescu
Abstract

This work is concerned with issues that arise in implementation and use of Support Vector Machines (SVM). First, an analytical computational approach to solve the convex optimization problem for SVM without any conventional heuristic optimization techniques is considered. The second issue concerns the application of SVM with linear kernels to data sets which are not linearly separable by approximating the separating surface with a collection of hyperplanes derived on small subsets of the training data. The simulation experiments on two-dimensional non-linear datasets result in good approximations of the true separating surface by the algorithm proposed. Finally, the non-linear transformation to obtain a non-linear separating surface by SVM is proposed and the piecewise linear approximation is extended to the piecewise non-linear approximation. The experimental results from both methods are compared.
Acknowledgements

First, I would like to express my gratitude to my research adviser Dr. Anca Ralescu, for her generous help and patient support. She motivated me a lot, she inspired me a lot.

To the members of my MS thesis committee, Dr. Raj K. Bhatnagar, Dr. Qing-An Zeng, and Dr. Dan Ralescu, my sincere thank you for their careful reading of my thesis.

I deeply thank Prof. Hitoshi Furuta for providing me with the opportunity to achieve this.

A special thanks to the lab members, Mircea Ionescu, Waibhav Tembe, and Sofia Visa, for their kind help!

Another special thanks goes to all of my friends, far and near, for their unconditional support and encouragements!

Finally, to my mother, father, sister, and aunt, thank you very much for their continuous support!
Contents

1 Introduction ................................................. 1

2 Background .................................................. 3
   2.1 Machine Learning ......................................... 3
   2.2 Types of Learning ......................................... 4
   2.3 Classification, Pattern Recognition ......................... 5

3 Support Vector Machines ..................................... 7
   3.1 Introduction ............................................... 7
   3.2 Definition of Hard Margin SVM: The Linearly Separable Case 9
      3.2.1 The Primal Form ........................................ 9
      3.2.2 The Dual Form ........................................ 10
   3.3 Definition of Soft Margin SVM: The Linearly Non-separable Case 12
      3.3.1 The Primal Form .................................... 13
      3.3.2 The Dual Form .................................... 14
   3.4 Kernel Functions: The Non-linear Case ...................... 15
   3.5 Implementing the SVM Approach ............................ 17

4 Solving SVM .................................................. 19
4.1 Quadratic Programming Problem for SVM ................................................. 19
4.2 Analytical Computational Approach ......................................................... 20

5 Piecewise Linear Approximation ................................................................. 33
  5.1 Iterative SVM algorithm ................................................................. 33
  5.2 Simulation Results .................................................................................. 35
    5.2.1 Linearly Separable Case ................................................................. 36
    5.2.2 Non-Linear Case 1 .......................................................................... 37
    5.2.3 Non-Linear Case 2 .......................................................................... 39
  5.3 Computational Aspects ............................................................................. 41
  5.4 Modeling versus Generalization Power ..................................................... 43

6 Piecewise Non-Linear Approximation ............................................................ 45
  6.1 Non-Linear Transformation in the Original Data Space ......................... 45
  6.2 Piecewise Non-Linear Approximation ..................................................... 48
  6.3 Simulation Results .................................................................................. 49
    6.3.1 Quadratic Approximation of Non-linear Separating Surfaces .. ...... 49
    6.3.2 Quadratic Approximation of the Cubic Separating Surface .......... 50
    6.3.3 Quadratic Approximation of the Sigmoid Separating Surface ...... 51
    6.3.4 Cubic Approximation of Non-linear Separating Surfaces .............. 51
  6.4 Conclusion ............................................................................................. 52

7 Conclusion and Future Work ........................................................................... 58
  7.1 Conclusion .............................................................................................. 58
  7.2 Future Work ........................................................................................... 59
    7.2.1 Selection of Training Subsets .......................................................... 59
7.2.2 Use of Other Kernels ...................................................... 60
7.2.3 Non-Linear Transformation of Attribute Axes .................. 60
7.2.4 Inference of the Final Separating Surface ...................... 60
7.2.5 Approximations of Difficult Class Boundaries ................. 61
# List of Figures

2.1 Difficult cases in classification .................................................. 6

3.1 Support vectors, Hyperplanes, Margin .......................................... 8

3.2 Data mapping by a kernel ................................................................. 16

4.1 The Basic Algorithm for finding the solution to the hard margin SVM .......... 20

4.2 Training data in Example 4.2.1 (linearly separable) ............................... 21

4.3 Training data in Example 4.2.2 (linearly non-separable) ......................... 23

4.4 Two hyperplanes generated in Example 4.2.2 .................................. 26

4.5 Training data in Example 4.2.2 (linearly non-separable) after mapping ........ 27

4.6 Separating surface in three dimensional space generated in Example 4.2.3 ....... 29

4.7 Training data in Example 4.2.2 (linearly non-separable) after kernel mapping ... 30

5.1 The Iterative SVM Algorithm for an arbitrary set ................................ 34

5.2 Two classes separated by a linear surface ....................................... 36

5.3 An example of the simulation results for the linearly separable data (Figure 5.2) with subset size 8 and distance threshold 11, 96.5% of classification accuracy for the test data ................................................................. 37

5.4 Two classes separated by a cubic curve ............................................ 38
5.5 An example of the simulation results for the polynomial data (Figure 5.4) with subset size 8 and distance threshold 7.2, 93.5% of classification accuracy for the test data

5.6 Two classes separated by a sigmoidal curve

5.7 An example of the simulation results for the sigmoid data (Figure 5.6) with subset size 8 and distance threshold 3.4, 83.0% of classification accuracy for the test data

5.8 The number of hyperplanes generated in training (x-axis) and the corresponding recognition rate for the test data (y-axis) with subset of size six in simulation 5.2.3

(100 runs)

5.9 The number of hyperplanes generated in training (x-axis) and the corresponding recognition rate for the test data (y-axis) with subset of size eight in simulation 5.2.3

(100 runs)

6.1 The Non-Linear Transformation for obtaining a non-linear separating surface by SVM

6.2 Simple example of the Non-Linear Transformation

6.3 The Iterative Non-Linear Approximation Algorithm

6.4 Quadratic approximation of the cubic separating surface in Figure 5.4 with subset size 8 and distance threshold 7.5.

6.5 Quadratic approximation of the sigmoidal separating surface in Figure 5.6 with subset size 6 and distance threshold 9.

6.6 Cubic approximation of the cubic separating surface in Figure 5.4 with subset size 8 and distance threshold 7.5.

6.7 Optimal curve approximation of the cubic separating surface in Figure 5.4 with subset size 6 and distance threshold 10.
## List of Tables

5.1 Simulation results for the linearly separable data in Figure 5.2 .......................... 37

5.2 Simulation results for the linear approximation of the cubic separating surface in
Figure 5.4 .................................................................................................................. 38

5.3 Simulation results for the linear approximation of the sigmoidal separating surface
in Figure 5.6 ............................................................................................................. 41

5.4 Comparison of the results from the original and the modified (one error per selected
hyperplane is allowed) ISVM .................................................................................. 44

6.1 Simulation results for the quadratic approximation of the cubic separating surface
in Figure 5.4. ............................................................................................................ 50

6.2 Simulation results for the quadratic approximation of the sigmoid separating surface
in Figure 5.6. .......................................................................................................... 51

6.3 Simulation results for the cubic approximation of the cubic separating surface in
Figure 5.4. .............................................................................................................. 52
Chapter 1

Introduction

Today, we live in a society with massive information, and the amount of information grows every day, every moment. Thanks to the advancing hardware technologies, the requirements for the memory storage to keep up with the increasing amount of data are quite satisfied. As a recent example, biologists have succeeded in identifying all the human genes and that information was stored into a small disk. Also, information such as pictures and movies can be stored as image data as itself.

What is now desired and also it is said as a big challenge for computer scientists is to find efficient algorithms to analyze such huge amount of data, extract important information from it, and furthermore, predict the future based on it. Although human beings do have such abilities and they are vital to live, it becomes impossible and this is why the requirement arises when the data is not perceptible to humans due to its large quantity and high dimensionality. Lots of researches for those problems have been ongoing in the Artificial Intelligence area [1], with inventing powerful techniques such as Fuzzy Inferences, Neural Networks, Bayesian Belief Networks, and Support Vector Machines.

This work concerns with issues that arise in the implementation and the use of Support Vector
Machines. The organization of this thesis is as follows: In chapter 2, the concepts of the machine learning, classification and pattern recognition, which have to be addressed as background knowledge for the further discussion, is described. Chapter 3 details the definition of Support Vector Machines and the problems lie in this technique. Chapter 4 illustrates an analytical computational approach to solve the convex optimization problem for Support Vector Machines without conventional heuristic optimization techniques. Chapter 5 describes the algorithm to approximate the actual separating surface with a collection of hyperplanes generated for the subsets of the training data iteratively. Experimental results and computational aspects are also included. The method to obtain a non-linear separating surface by Support Vector Machines and the extended algorithm are introduced in chapter 6 along with several simulation results. Finally, chapter 7 concludes this thesis with future works.
Chapter 2

Background

2.1 Machine Learning

Learning is one of the most significant abilities that humans have, yet computers do not. Humans seem to obtain general concepts through specific experience or even through intentional training and use them to adjust themselves to the environment and to take appropriate actions for situation even when this situation is new. Imagine that John has just moved to a new town. John might want to walk around to know the surroundings better. For the first time, John might get lost, or he might have to take a long walk to get to the nearest grocery store. Then, what happens after a week later? Ideally, John knows the shortest path to the grocery store and might even know a nice coffee shop. Since people had started to consider computers as potential tools, it has been desired to find efficient algorithms to learn general concepts and improve their performance from experience by itself like humans do.

In recent years, many successful machine learning applications for particular tasks have been developed in various fields [1] [2]. For example, as pattern recognition methods, text recognition and speech recognition systems have revealed better performance than traditional approaches.
In data mining field, lots of commercial applications play important roles as powerful business strategies to discover the tendency of the market or analyze the preference of customers from huge database. Medical diagnosis systems which obtain knowledge from medical records and return effective treatments to the diseases are proposed. Similarly in bioinformatics, machine learning techniques are commonly used. Also in the control field, automated driving systems are expected to become for a practical use in the near future.

2.2 Types of Learning

From the point of the learning process, there are two types of learning, *supervised* and *unsupervised* learning.

In supervised learning, the system evolves based on the training data. Training data consist of input/output pairs which are the desired responses to the system. This is exactly like children learning characters by repeatedly copying model letters written by a teacher. Hence, when we have enough data in which sufficient quality and fair distribution can be expected, supervised learning is a good option to train the system. However, the selection of the training set has to be done very carefully because it determines completely the performance of the resulting system. This is just as trying to find a good teacher for your children. The best known example of this type of learning is the backpropagation algorithm, a typical training model for Neural Networks. In the backpropagation algorithm, the error between the desired outputs (that is, the training set) and those from the network is computed and “backpropagated” to the network, then the network is configured by reducing the error. The Support Vector Machines, the subject matter of this paper is a relatively new technique of supervised learning.

In unsupervised learning, the training data does not contain any distinction between input and
output pairs, that is, the system is not provided any information about the desired output. The goal of this learning is to build representations from the data and take appropriate reactions to an environment which does not have the exact training set. This is known as self-organization or adaptation. Unsupervised learning is not well understood although it is one of the typical learning methods of humans and this is why researches in this area are attracting lots of attention.

2.3 Classification, Pattern Recognition

The typical problems where the machine learning techniques exhibit their remarkable powers most are in the field of Pattern Recognition. We view Pattern Recognition as Classification: assign an input object $X$ into a category, let’s say, $class_1$ or $class_2$ or ... or $class_M$, depending on the features that characterize the classes.

An object $X$ and the classes are abstractly represented by some sets of features as a vector which can be seen as a point in a feature space where the similarities are measured.

The most intuitive classifier, called minimum distance classifier, computes the distance in the feature space between the input $X$ and each class (or separation of it), then classifies $X$ into the class with minimum distance. The distance (can be Euclidean distance, Manhattan distance, Mahalanobis distance, etc) is used as a similarity measure and its selection affects the shapes of the class region.

The difficulty in classification problems occurs when the input belonging to different classes have some correlations, or when wide variety can be expected within one class. Especially, these cases shown in Figure 2.1 are said to be difficult to deal with by the simple minimum distance classifier.

Using a Neural Network is the most popular strategy for these problems, and in fact, in many
cases, Neural Networks have shown very good results. However, several shortcomings arise in that constructing a Neural Network requires many problem-dependent parameters which must be decided by trial and error. Also, even if a network which returns ideal output is obtained, the mechanism of the network, actually, the mechanism of the object represented by the network is only implicitly explored by the network and hard to extract from it.

Figure 2.1: Difficult cases in classification
Chapter 3

Support Vector Machines

3.1 Introduction

Since Support Vector Machine (SVM) was introduced by Vapnik [3] [4], it has emerged as one of the most interesting and potentially powerful classification techniques [5] [6] [7] [8]. Many successful applications using SVM can be found in various fields, such as bioinformatics, handwritten character recognition, text categorization, object recognition, and speech recognition.

If the training data is linearly separable, an infinite number of separating surfaces which correctly classify the data exist. SVM is a classifier which determines a linear separating surface, hyperplane, so that the distance between the hyperplane and the training datum nearest to the hyperplane is maximized. In other words, SVM finds the hyperplane with maximum generalization ability. The data points closest to the separating hyperplane are called support vectors, while the distance from these to the hyperplane is called margin. Figure 3.1 illustrates the concepts described above.

Different from Neural Network classifiers, SVM is deterministic on the training data. Furthermore, the optimal separating hyperplane is decided depending only on the support vectors. This
fact can be utilized to reduce the number of training data actually used for the training and improve
the training effect [9] [10].

There are two types of linear version of SVM: the hard margin SVM and the soft margin SVM. The soft margin SVM introduces the slack variables in order to deal with noisy data. Section 3.2
and 3.3 describe these two methods, respectively.

In practice, however, one can hardly expect linear separation over arbitrary data. SVM provides
a powerful technique that can be extended to a nonlinear classifier by using kernel functions. Kernel
functions perform a data mapping into a higher dimensional feature space, and after this operation,
the data points are expected to become linearly separable or close to linearly separable in the new
feature space. While several classes of “good” kernel functions are proposed in literature and new
kernel function can be generated easily, there is no general theory for kernel selection. Kernel
functions are described in section 3.4.
Since the resulting surface obtained by SVM is always a linear function, multi-class problem are considered based on the simplest case, that is, two-class problem. The extension from two-class problem to multi-class problem is one of the major research areas of SVM. Several methods have been proposed, such as decision tree formulation, one-to-one formulation, and one-against-all formulation [5]. The research presented in this thesis focuses on the two-class problem.

3.2 Definition of Hard Margin SVM: The Linearly Separable Case

3.2.1 The Primal Form

For a two-class, linearly separable case, given the \( m \)-dimensional input data \( (x_i, y_i)(i = 1, \ldots, M) \) that belong to \( \text{class}_1 \) if \( y_i = +1 \) or \( \text{class}_2 \) if \( y_i = -1 \), the separating hyperplane, \( D(x) \), of the two classes is given by

\[
D(x) = w^t x + b
\]  

(3.1)

where \( w \) is a \( m \)-dimensional vector and \( b \) is a bias. The training data must satisfy the constraints,

\[
w^t x_i + b \geq +1, \quad \text{for} \quad y_i = +1
\]  

(3.2)

\[
w^t x_i + b \leq -1, \quad \text{for} \quad y_i = -1
\]  

(3.3)

which can be rewritten in a condensed form as

\[
y_i(w^t x_i + b) \geq 1, \quad i = 1, \ldots, M
\]  

(3.4)
The data points which satisfy the equality in either (3.2) or (3.3) are support vectors, and they
lie on the hyperplane $H_1 : wx_i + b = 1$ and $H_2 : wx_i + b = -1$, respectively. SVM determines the
optimal separating hyperplane $H : wx_i + b = 0$ which maximizes the generalization region,$\{x | -1 \leq D(x) \leq 1\}$. This region is called margin and it is determined by the Euclidean distance $d_1$ from the
hyperplane $H_1$ to the optimal hyperplane $H$, plus the Euclidian distance $d_2$ from the hyperplane
$H_2$ to $H$, that is, $d_1 + d_2$. Since $d_1 = d_2 = 1/\|w\|$, the margin is $2/\|w\|$. Therefore, the optimal
separating hyperplane with the maximum margin is obtained by solving

$$\text{Minimize: } \frac{1}{2}\|w\|^2$$

with respect to $w$ and $b$, subject to the constraints (3.4).

The convex optimization problem given by (3.5) and (3.4) refers to the primal form, and can
be solved by the quadratic programming technique when the number of input data is small.

3.2.2 The Dual Form

Introducing positive Lagrange multipliers $\alpha_i$ ($i = 1, \ldots, M$), the constrained problem given by (3.5)
and (3.4) is converted into the unconstrained problem:

$$Q(w, b, \alpha) = \frac{1}{2}w^Tw - \sum_{i=1}^{M} \alpha_i\{y_i(w^Tx_i + b) - 1\}$$

(3.6)

The optimal solution of (3.6) is obtained by minimizing $Q(w, b, \alpha)$ with respect to $w$, $b$, and max-
imizing with respect to $\alpha_i (\alpha_i \geq 0)$. In addition, the optimal solution $(w, b, \alpha)$ must satisfy the
following conditions:

$$\frac{\delta Q(w, b, \alpha)}{\delta b} = 0$$

(3.7)
\[ \frac{\delta Q(w, b, \alpha)}{\delta w} = 0 \]  

(3.8)

Using (3.6), (3.7) becomes

\[ \sum_{i=1}^{M} \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad \text{for } i = 1, \ldots, M \]  

(3.9)

and (3.8) becomes

\[ w = \sum_{i=1}^{M} \alpha_i y_i x_i, \quad \alpha_i \geq 0, \quad \text{for } i = 1, \ldots, M \]  

(3.10)

Substituting (3.9) and (3.10) into (3.6), we obtain the dual formulation, that is

\[ \text{Maximize : } Q(\alpha) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j x_i^T x_j \]  

(3.11)

with respect to \( \alpha_i \) subject to the constraints (3.12) and (3.13).

\[ \sum_{i=1}^{M} \alpha_i y_i = 0 \]  

(3.12)

\[ \alpha_i \geq 0, \quad \text{for } i = 1, \ldots, M \]  

(3.13)

Also, it is known from optimization theory [6] [7] that the solution to this optimization problem must verify the Karush-Kuhn-Tucker (KKT) conditions:

\[ \alpha_i \{ y_i (w^T x_i + b) - 1 \} = 0 \]  

(3.14)
The SVM formalized from (3.11) - (3.14) is called hard margin SVM. Solving for $a_i$, we obtain the support vectors which are the data points corresponding to $a_i \neq 0$. The separating hyperplane is given by

$$D(x) = \sum_{i=1}^{M} a_i y_i x_i^T x + b \quad (3.15)$$

where

$$b = y_i - w^T x_i, \text{ for all } a_i \neq 0 \quad (3.16)$$

Classification of a data point is done according to the sign of the hyperplane evaluated at that point, that is, $x$ is classified into $\text{class}_1$ if $D(x) > 0$ and into $\text{class}_2$ if $D(x) < 0$.

**Why Dual Form?**

Duality is a crucial property of SVM. As seen in the dual objective function (3.11), the training data only appear inside the dot products. This is important when the extension of linear version of SVM into non-linear case is needed. The extension can be done by making use of a kernel function, and with dual form, it is an easy procedure, in fact, just a plug-in operation. Formal description of this extension is discussed in section 3.4.

### 3.3 Definition of Soft Margin SVM: The Linearly Non-separable Case

When the training data is considered to be non-separable due to some noise, that is, the inequality constraints (3.2) and (3.3) do not hold for some data, soft margin SVM adopt slack variables
to “soften” the constraint which the optimal hyperplane should satisfy, then it determines the hyperplane with both the maximum margin and the minimum classification error.

3.3.1 The Primal Form

To derive the primal form, the nonnegative slack variables $\xi_i (> 0), i = 1, \ldots, M$ are introduced in (3.4) in order to allow the data that do not have the maximum margin to exist,

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, M$$  \hspace{1cm} (3.17)

Suppose, as an example, the training data $x_i$ belongs to class 1. If $0 < \xi_i < 1$, $x_i$ is correctly classified even though it does not have the maximum margin. It lies between the hyperplane $H_1 : wx_i + b = 1$ and the optimal hyperplane $H : wx_i + b = 0$. Else, if $\xi_i \geq 1$, $x_i$ is misclassified and it lies beyond the optimal hyperplane $H$ toward another hyperplane $H_2 : wx_i + b = -1$.

Now, the goal of the soft margin SVM is to find a separating hyperplane which can minimize the classification error while maximizing the margin between the classes. Therefore, the primal form of soft margin SVM is formalized as solving

$$\text{Minimize}: \frac{1}{2}||w||^2 + C \sum_{i=1}^{M} \xi_i$$  \hspace{1cm} (3.18)

subject to the constraints (3.17) and

$$\xi_i \geq 0, \quad \text{for} \quad i = 1, \ldots, M$$  \hspace{1cm} (3.19)

where the parameter $C > 0$ controls the tradeoff between the minimization of classification error and maximization of margin. The larger $C$ is set to, the smaller the number of misclassification
expected, while the smaller $C$ is , the wider margin would be.

### 3.3.2 The Dual Form

Similar to the hard margin SVM, Lagrange multipliers are introduced to obtain the dual formulation. Here, two different multipliers, $\alpha$ and $\beta$ for the constraints (3.17) and (3.19) are needed. Then, (3.18), (3.17) and (3.19) becomes,

$$Q(w,b,\xi,\alpha,\beta) = \frac{1}{2}w^Tw + C \sum_{i=1}^{M} \xi_i - \sum_{i=1}^{M} \alpha_i \{y_i (w^T x_i + b) - 1 + \xi_i\} - \sum_{i=1}^{M} \beta_i \xi_i \quad (3.20)$$

The optimal solution $(w,b,\xi,\alpha,\beta)$ must satisfy are

$$\frac{\delta Q(w,b,\xi,\alpha,\beta)}{\delta b} = 0 \quad (3.21)$$

$$\frac{\delta Q(w,b,\xi,\alpha,\beta)}{\delta w} = 0 \quad (3.22)$$

$$\frac{\delta Q(w,b,\xi,\alpha,\beta)}{\delta \xi} = 0 \quad (3.23)$$

From (3.20), these become, respectively,

$$\sum_{i=1}^{M} \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad \text{for} \quad i = 1,\ldots,M \quad (3.24)$$

$$w = \sum_{i=1}^{M} \alpha_i y_i x_i, \quad \alpha_i \geq 0, \quad \text{for} \quad i = 1,\ldots,M \quad (3.25)$$
\[ \alpha_i + \beta_i = C, \quad \alpha_i, \beta_i \geq 0, \quad \text{for } i = 1, \ldots, M \]  \hspace{1cm} (3.26)

Substituting (3.24) to (3.26) into (3.20), the dual formulation is obtained:

\[ \text{Maximize : } Q(\alpha) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j x_i^T x_j \]  \hspace{1cm} (3.27)

subject to

\[ \sum_{i=1}^{M} \alpha_i y_i = 0 \]  \hspace{1cm} (3.28)

\[ 0 \leq \alpha_i \leq C, \quad \text{for } i = 1, \ldots, M \]  \hspace{1cm} (3.29)

Also, the optimal solution must satisfy the KKT conditions:

\[ \alpha_i \{ y_i (w^T x_i + b) - 1 + \xi_i \} = 0 \]  \hspace{1cm} (3.30)

\[ \beta_i \xi_i = (C - \alpha_i) \xi_i = 0 \]  \hspace{1cm} (3.31)

The separating hyperplane is the same for the hard margin SVM which is given by (3.15).

### 3.4 Kernel Functions: The Non-linear Case

When the optimal separating surface for the data is non-linear, SVM first map the input data by \( \phi \) into a higher dimensional feature space where they are expected to be linearly separable. Then,
the problem can be solved in the same fashion as the linear version in the new feature space (Figure 3.2).

![kernel mapping diagram](image)

**Figure 3.2: Data mapping by a kernel**

A kernel \( K(x_i, x) \) is a function that performs the mapping such that the following relation holds:

\[
K(x_i, x_j) = \phi(x_i)^T \phi(x_j)
\]  

(3.32)

There are several common types of kernels such as polynomial kernels and radial basis function kernels. Other kernels can be constructed as long as the Mercer’s theorem [6] is satisfied. Further discussion of Mercer’s theorem is beyond the scope of this research.

In the dual representation, as shown in section 3.2.2 and 3.3.2, the input data only appear inside the dot products. Therefore, by simply performing the following substitution in (3.11) or (3.27),

\[
x_i^T x_j \leftarrow K(x_i, x_j) = \phi(x_i)^T \phi(x_j)
\]  

(3.33)
The optimization problem to be solved becomes

\[
\text{Maximize} : \quad Q(\alpha) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j K(x_i, x_j)
\]  \quad (3.34)

under the same constraints from (3.12) to (3.14) for the hard margin SVM, or from (3.28) to (3.31) for the soft margin SVM. Similarly, from (3.15) and (3.16), the separating surface is given by

\[
D(x) = \sum_{i=1}^{M} \alpha_i y_i K(x_i, x) + b
\]  \quad (3.35)

where

\[
b = y_i - \sum_{i=1}^{M} \alpha_i y_i K(x_i, x), \quad \text{for all} \quad \alpha \neq 0
\]  \quad (3.36)

The separating surface (3.35) is indeed a linear function in the new feature space. The data \(x\) is classified according to the sign of (3.35) in the feature space.

### 3.5 Implementing the SVM Approach

Training the SVM, equivalent to solve the convex optimization problem defined in section 3.2 and 3.3, is actually a quadratic programming problem with the number of variables equal to the number of training data. Though this type of problem is well understood, conventional heuristic techniques become infeasible because of their memory and time requirements for a large training set. Widely used implementation techniques in current SVM research, Sequential Minimal Optimization (SMO) [6] [11] and \(SV M^{light}\) [12] are based on solving the quadratic programming problem posed by the SVM. They differ in their ways of chunking it into smaller quadratic programming problems. An analytical computational approach to solve the optimization problem for SVM without any
conventional heuristic optimization techniques is considered in chapter 4.

As already mentioned, although utilizing a kernel function for the non-linear case is one of the most significant property of SVM and the performance of SVM strongly depends on it, theories about the selection of the best kernel function for the corresponding training data are not known. Even when, by trial and error, a good kernel function is obtained, working in a high dimensional space with large vectors introduces a computational problem [13]. It becomes then interesting to investigate to what extent the linear version of SVM can be used for problems where linear separability fails to approximate the real separating surface. The algorithm proposed in chapter 5 uses only the hard margin SVM iteratively and the optimal separating surface consists of a combination of hyperplanes obtained for randomly selected small subsets of the training data.
Chapter 4

Solving SVM

4.1 Quadratic Programming Problem for SVM

In an optimization problem, the types of mathematical relationship between the objective function and the constraints determine the methods that can be applied for the optimization. For example, an optimization problem which has a linear objective function and linear constraints is called Linear Programming (LP) Problem, and usually solved with the Simplex method or Newton-Barrier method.

A Quadratic Programming (QP) Problem is defined as an optimization problem to optimize a quadratic function subject to linear equality/inequality constraints. As shown in chapter 3, finding an optimal separating hyperplane by SVM is a QP problem. This type of problems is well understood and several heuristic techniques such as Generalized Reduced Gradient (GRG) and Sequential Quadratic Programming (SQP) are known to achieve excellent approximation to the optimal solution. However, several issues arise in designing a SVM learner. Since it requires to keep a matrix of which the size is the square of the number of training data through the training (see equation (3.11)), conventional heuristic techniques quickly become impracticable in their memory
and time requirements for a large training data. One common strategy to overcome these issues is to break the entire QP problem into a series of smaller QP problems [14]. The approach considered in the following sections also exploits Osuna’s idea for the SMO algorithm [11], however, the main discussion here lies in finding the solution for the small QP problem analytically.

### 4.2 Analytical Computational Approach

The analytical solution for the “small” QP problem for SVM proceeds from the observation that the KKT conditions are necessary for the optimal solution of any convex optimization problem [6] [7]. Accordingly, the steps taken to find the optimal solution are described in the Basic Algorithm of Figure 4.1.

```
BA1. Solve for $\alpha_i, i = 1, \ldots, M$ and $b$, the KKT conditions (3.14) and (3.12).

BA2. Eliminate from the solutions obtained at BA1, those which do not satisfy (3.13).

BA3. Select from the remaining candidates those which maximize $Q(\alpha)$.
```

Figure 4.1: The Basic Algorithm for finding the solution to the hard margin SVM

The procedure described above corresponds to the hard margin SVM defined in section 3.2. This procedure will output the correct SVM solution (maximum margin separating hyperplane) when the data set is linearly separable. However, when this is not the case, that is, when there is no separating hyperplane, the procedure may still output a solution. Therefore, when it is not known whether the data set is linearly separable it is also not known if the resulting hyperplane should be accepted. Since, in the linearly separable case the SVM solution is unique obtaining more than one
hyperplane in step BA3 would indicate that the data set is not linearly separable. The examples below illustrate this discussion.

Example 4.2.1 (Linearly Separable Case)

This example illustrates all the steps to solve a simple case by following the Basic Algorithm, with linearly separable, one-dimensional training data (Figure 4.2).

![Diagram of linearly separable data](image.png)

*Figure 4.2: Training data in Example 4.2.1 (linearly separable)*

The objective function $Q(\alpha)$ for this problem is, $Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2}(\alpha_1 - \alpha_3)^2$.

At step BA1, equations (1)-(4) below are solved. Equations (1), (2), (3) are obtained from the KKT conditions of (3.14) for these three training data points; (4) is the constraint from (3.12).

1. \( \alpha_1(\alpha_1 + \alpha_3 + b - 1) = 0 \)
2. \( \alpha_2(-b - 1) = 0 \)
3. \( \alpha_3(\alpha_1 + \alpha_3 - b - 1) = 0 \)
4. \( \alpha_1 - \alpha_2 - \alpha_3 = 0 \)

There are eight possible cases and the possible solutions as follows.
(a) $\alpha_2 = 0, \alpha_1 = 0, \alpha_3 = 0$

$\alpha_1 = \alpha_2 = \alpha_3 = 0$

$b : undetermined$

(b) $\alpha_2 = 0, \alpha_1 = 0, \alpha_1 + \alpha_3 - b - 1 = 0$

$\alpha_1 = \alpha_2 = \alpha_3 = 0$

$b = -1$

(c) $\alpha_2 = 0, \alpha_3 = 0, \alpha_1 + \alpha_3 + b - 1 = 0$

$\alpha_1 = \alpha_2 = \alpha_3 = 0$

$b = 1$

(d) $\alpha_2 = 0, \alpha_1 + \alpha_3 + b - 1 = 0, \alpha_1 + \alpha_3 - b - 1 = 0$

$\alpha_1 = 1/2, \alpha_2 = 0, \alpha_3 = 1/2$

$b = 0$

(e) $b = 1, \alpha_1 = 0, \alpha_3 = 0$

$\alpha_1 = \alpha_2 = \alpha_3 = 0$

$b = 1$

(f) $b = 1, \alpha_1 = 0, \alpha_1 + \alpha_3 - b - 1 = 0$

$\alpha_1 = \alpha_2 = \alpha_3 = 0$

$b = -1$

(g) $b = 1, \alpha_3 = 0, \alpha_1 + \alpha_3 + b - 1 = 0$
\[ \alpha_1 = 2, \alpha_2 = 2, \alpha_3 = 0 \]
\[ b = -1 \]

(h) \( b = 1, \alpha_1 + \alpha_3 + b - 1 = 0, \alpha_1 + \alpha_3 - b - 1 = 0 \)

\[ \alpha_1, \alpha_2, \alpha_3 : \text{undetermined} \]
\[ b = -1 \]

At step BA2, all solutions except (h) from BA1 satisfy the constraint \( \alpha_i \geq 0 \) (from (3.13)).

Since the solution from (g) maximizes \( Q(\alpha) \) at step BA3, \( \alpha_1 = 2, \alpha_2 = 2, \alpha_3 = 0, b = -1 \) is the optimal solution. Therefore, \((-1, 0)\) and \((0, 0)\) are support vectors and the optimal separating hyperplane is given by the equation \( x_1 = -1/2 \). In fact, this hyperplane correctly classifies all the input data with having maximum margin as well.

**Example 4.2.2 (Linera Non-separable Case)**

In this example, the Basic Algorithm is applied for linearly non-separable, one-dimensional training data shown in Figure 4.3.

```
<table>
<thead>
<tr>
<th>Class 1</th>
<th>Class 2</th>
<th>Class 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-1, 0)</td>
<td>(0, 0)</td>
<td>(1, 0)</td>
</tr>
<tr>
<td>X1</td>
<td>X2</td>
<td>X3</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>+1</td>
<td>class3</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>class2</td>
</tr>
<tr>
<td>1</td>
<td>+1</td>
<td>class1</td>
</tr>
</tbody>
</table>
```

*Figure 4.3: Training data in Example 4.2.2 (linearly non-separable)*

The objective function \( Q(\alpha) \) for this problem is, \( Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2}(\alpha_1 - \alpha_3)^2 \).
At step BA1, equations (1)-(4) below are solved. Equations (1), (2), (3) are obtained from the KKT conditions of (3.14) for these three training data points; (4) is the constraint from (3.12).

(1) \( \alpha_1(\alpha_1 - \alpha_3 + b - 1) = 0 \)

(2) \( \alpha_2(-b - 1) = 0 \)

(3) \( \alpha_3(-\alpha_1 + \alpha_3 + b - 1) = 0 \)

(4) \( \alpha_1 - \alpha_2 + \alpha_3 = 0 \)

There are eight possible cases and the possible solutions as follows.

(a) \( \alpha_2 = 0, \alpha_1 = 0, \alpha_3 = 0 \)

\[ \alpha_1 = \alpha_2 = \alpha_3 = 0 \]

\( b: \text{undetermined} \)

(b) \( \alpha_2 = 0, \alpha_1 = 0, -\alpha_1 + \alpha_3 + b - 1 = 0 \)

\[ \alpha_1 = \alpha_2 = \alpha_3 = 0 \]

\( b = 1 \)

(c) \( \alpha_2 = 0, \alpha_3 = 0, \alpha_1 - \alpha_3 + b - 1 = 0 \)

\[ \alpha_1 = \alpha_2 = \alpha_3 = 0 \]

\( b = 1 \)

(d) \( \alpha_2 = 0, \alpha_1 - \alpha_3 + b - 1 = 0, -\alpha_1 + \alpha_3 + b - 1 = 0 \)

\[ \alpha_1 = \alpha_2 = \alpha_3 = 0 \]

\( b = 1 \)

(e) \( b = -1, \alpha_1 = 0, \alpha_3 = 0 \)
\[ \alpha_1 = \alpha_2 = \alpha_3 = 0 \]
\[ b = -1 \]

(f) \[ b = -1, \alpha_1 = 0, -\alpha_1 + \alpha_3 + b - 1 = 0 \]
\[ \alpha_1 = 0, \alpha_2 = 2, \alpha_3 = 2 \]
\[ b = -1 \]

(g) \[ b = -1, \alpha_3 = 0, \alpha_1 - \alpha_3 + b - 1 = 0 \]
\[ \alpha_1 = 2, \alpha_2 = 2, \alpha_3 = 0 \]
\[ b = -1 \]

(h) \[ b = -1, \alpha_1 - \alpha_3 + b - 1 = 0, -\alpha_1 + \alpha_3 + b - 1 = 0 \]
\[ \alpha_1, \alpha_2, \alpha_3 : \text{undetermined} \]
\[ b = -1 \]

At step BA2, all solutions except (h) from BA1 satisfy the constraint \( \alpha_i \geq 0 \) (from (3.13)).

The solutions (f) and (g) shown in Figure 4.4, maximize \( Q(\alpha) \) at BA3. The hyperplane obtained according to the solution (f) is given by \( D_1(x_1, x_2) = 2x_1 - x_2 - 1 \) having \((0, 0)\) and \((1, 0)\) as support vectors. Similarly, the hyperplane obtained for (g) is given by \( D_2(x_1, x_2) = -2x_1 - x_2 - 1 \) having \((-1, 0)\) and \((0, 0)\) as support vectors. Neither hyperplane by itself separates the classes correctly. However, the polygonal line \( D(x_1, x_2) \), defined in (4.1) and obtained from combining them, does.

\[
D(x_1, x_2) = \begin{cases} 
D_2(x_1, x_2) & \text{if } x_1 < 0 \\
D_1(x_1, x_2) & \text{otherwise}
\end{cases} \quad (4.1)
\]
This observation is important for the subsequent developments in this thesis, as it suggests that the nonlinear separating surface can be composed of pieces of linear surfaces.

Example 4.2.3 (Linearly Non-separable Case with Data Mapping)

This example illustrates how the Basic Algorithm works when the linearly non-separable data is first mapped into a higher dimensional space. The linearly non-separable training data from the previous example (in Figure 4.3) is mapped into the three dimensional space by

\[(x_1, x_2) \rightarrow (x_1, x_2, x_1^2)\]  \hspace{1cm} (4.2)

After this mapping, the training data become as shown in Figure 4.5. The Basic Algorithm is applied for the mapped data in the same way as previous examples. The objective function \(Q(\alpha)\) for this problem is, \(Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 - (\alpha_1^2 + \alpha_3^2)\).

At step BA1, equations (1)-(4) below are solved. Equations (1), (2), (3) are obtained from the
Figure 4.5: Training data in Example 4.2.2 (linearly non-separable) after mapping

KKT conditions of (3.14) for these three training data points; (4) is the constraint from (3.12).

1. \( \alpha_1 (2\alpha_1 + b - 1) = 0 \)
2. \( \alpha_2 (-b - 1) = 0 \)
3. \( \alpha_3 (2\alpha_3 + b - 1) = 0 \)
4. \( \alpha_1 - \alpha_2 + \alpha_3 = 0 \)

There are eight possible cases and the possible solutions as follows.

(a) \( \alpha_2 = 0, \alpha_1 = 0, \alpha_3 = 0 \)

\[ \alpha_1 = \alpha_2 = \alpha_3 = 0 \]

\( b : \text{undetermined} \)

(b) \( \alpha_2 = 0, \alpha_1 = 0, 2\alpha_3 + b - 1 = 0 \)

\[ \alpha_1 = \alpha_2 = \alpha_3 = 0 \]

\( b = 1 \)
(c) $\alpha_2 = 0, \alpha_3 = 0, 2\alpha_1 + b - 1 = 0$

\[
\alpha_1 = \alpha_2 = \alpha_3 = 0
\]

\[
b = 1
\]

(d) $\alpha_2 = 0, 2\alpha_1 + b - 1 = 0, 2\alpha_3 + b - 1 = 0$

\[
\alpha_1 = \alpha_2 = \alpha_3 = 0
\]

\[
b = \text{undetermined}
\]

(e) $b = -1, \alpha_1 = 0, \alpha_3 = 0$

\[
\alpha_1 = \alpha_2 = \alpha_3 = 0
\]

\[
b = -1
\]

(f) $b = -1, \alpha_1 = 0, 2\alpha_3 + b - 1 = 0$

\[
\alpha_1 = 0, \alpha_2 = 1, \alpha_3 = 1
\]

\[
b = -1
\]

(g) $b = -1, \alpha_3 = 0, 2\alpha_1 + b - 1 = 0$

\[
\alpha_1 = 1, \alpha_2 = 1, \alpha_3 = 0
\]

\[
b = -1
\]

(h) $b = -1, 2\alpha_1 + b - 1 = 0, 2\alpha_3 + b - 1 = 0$

\[
\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = 1
\]

\[
b = -1
\]

At step BA2, all solutions from BA1 satisfy the constraint $\alpha_i \geq 0$ (from (3.13)).
The solution from (h) maximizes $Q(\alpha)$ at BA3. The hyperplane obtained according to the solution (h) is given by $D(x_1, x_2, x_3) = 2x_3 - 1$, actually, it is a surface in the three dimensional space (Figure 4.6). All three points are support vectors in this example.

![Figure 4.6: Separating surface in three dimensional space generated in Example 4.2.3](image)

**Example 4.2.4 (Linearly Non-separable Case with Kernel Mapping)**

This example illustrates Basic Algorithm when the non-linear transformation 4.3 is applied to the training points. Unlike the transformation in example 4.2.3 the current one corresponds to the kernel mapping.

$$(x_1, x_2) \rightarrow \phi(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

After this mapping, the training data becomes linearly separable as shown in Figure 4.7. The Basic Algorithm is applied for the mapped data in the same way as previous examples. The objective function $Q(\alpha)$ for this problem is, $Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2}(\alpha_1^2 + \alpha_2^2)$.

At step BA1, equations (1)-(4) below are solved. Equations (1), (2), (3) are obtained from the KKT conditions of (3.14) for these three training data points; (4) is the constraint from (3.12).
Figure 4.7: Training data in Example 4.2.2 (linearly non-separable) after kernel mapping

(1) $\alpha_1(\alpha_1 + \alpha_3 + b - 1) = 0$

(2) $\alpha_2(-b - 1) = 0$

(3) $\alpha_3(\alpha_1 + \alpha_3 + b - 1) = 0$

(4) $\alpha_1 - \alpha_2 + \alpha_3 = 0$

There are eight possible cases and the possible solutions as follows.

(a) $\alpha_2 = 0, \alpha_1 = 0, \alpha_3 = 0$

$\alpha_1 = \alpha_2 = \alpha_3 = 0$

$b: \text{undetermined}$

(b) $\alpha_2 = 0, \alpha_1 = 0, \alpha_1 + \alpha_3 + b - 1 = 0$

$\alpha_1 = \alpha_2 = \alpha_3 = 0$

$b = 1$

(c) $\alpha_2 = 0, \alpha_3 = 0, \alpha_1 + \alpha_3 + b - 1 = 0$
\[ \alpha_1 = \alpha_2 = \alpha_3 = 0 \]
\[ b = 1 \]

(d) \[ \alpha_2 = 0, \alpha_1 + \alpha_3 + b - 1 = 0 \]
\[ \alpha_1, \alpha_3 : \text{undetermined}, \alpha_2 = 0 \]
\[ b = -1 \]

(e) \[ b = -1, \alpha_1 = 0, \alpha_3 = 0 \]
\[ \alpha_1 = \alpha_2 = \alpha_3 = 0 \]
\[ b = -1 \]

(f) \[ b = -1, \alpha_1 = 0, \alpha_1 + \alpha_3 + b - 1 = 0 \]
\[ \alpha_1 = 0, \alpha_2 = 2, \alpha_3 = 2 \]
\[ b = -1 \]

(g) \[ b = -1, \alpha_3 = 0, \alpha_1 + \alpha_3 + b - 1 = 0 \]
\[ \alpha_1 = 2, \alpha_2 = 2, \alpha_3 = 0 \]
\[ b = -1 \]

(h) \[ b = -1, \alpha_1 + \alpha_3 + b - 1 = 0 \]
\[ \alpha_1, \alpha_3 : \text{undetermined}, \alpha_2 = 2 \]
\[ b = -1 \]

At step BA2, all solutions except (d) and (h) satisfy the constraint \( \alpha_i \geq 0 \) (from (3.13)).

The solution from (f) and (g) maximizes \( Q(\alpha) \) at BA3. The hyperplane obtained according to the solutions (f) and (g) are both given by \( D(x_1, x_2, x_3) = 2x_1 - 1 \), that is, a line given by \( x_1 = 1/2 \).
The Basic Algorithm described in Figure 4.1 finds the optimal solution under the assumption that the input data is linearly separable. When this assumption holds, only the possible candidates which satisfy all the necessary constraints for the optimal solution remain after the step $BA_2$, and the one which maximizes the objective function $Q(\alpha)$ is selected as the optimal solution at step $BA_3$. Although it returns a solution even when the data is not linearly separable, the resulting hyperplane does not separate the two classes. Therefore, if the linear separability of the data is not known in advance, it is necessary to test whether the resulting hyperplane classifies all the input data correctly. As shown in the above examples, the computation at each step is done analytically and in each case the exact solution is obtained.
Chapter 5

Piecewise Linear Approximation

5.1 Iterative SVM algorithm

In the light of the previous chapters, the current section considers the problem of approximating a non-linear separating surface by using only the linear version SVM, (in fact, here, the hard margin SVM), without mapping into a higher dimensional feature space [15]. The resulting separating surface consists of the combination of hyperplanes [16] which are obtained for (randomly) selected small subsets of the training data and are optimal for the corresponding subsets. The algorithm ISVM shown in Figure 5.1 accomplishes this.

In ISVM1, the random selection of the training subsets can be constrained by various conditions. The idea behind constraining the selection of small subsets of the training data is to ensure that these are from around the boundary between the two classes. This idea appears in other work as well [9] [10] [17]. In the current implementation, a constraint on the distance between the two classes, as represented in this subset, is enforced. In terms of the SVM result this means that the hyperplanes generated at the step ISVM2 use training data points close to the true support vectors. Two cases are considered in ISVM2 where the Basic Algorithm is applied:
**ISVM1.** Constrained-random selection of subsets of training data.

**ISVM2.** Apply the *Basic Algorithm* to the current training set to obtain the corresponding optimal hyperplane.
(a) If the subset is linearly separable,
   keep the hyperplane generated for it.
   Do not return the subset to the training set.
(b) If the subset is not linearly separable,
   discard the hyperplane generated for it.
   Return it to the training set.

**ISVM3.** Repeat ISVM1 and ISVM2 until no more subsets can be generated.

**ISVM4.** Infer the overall separating surface by combining the hyperplanes obtained at the previous steps.

Figure 5.1: The Iterative SVM Algorithm for an arbitrary set

1. If the training subset generated in step ISVM1 is linearly separable the hyperplane produced by the *Basic Algorithm* is actually (at least) locally optimal. More precisely, based on the current training subset, it ensures minimum classification errors for any other points from the same region (determined by the range of the $x_1$ and $x_2$ coordinates in the training subset). Linearly separable training subsets are eliminated from further training.

2. If the training subset generated at step ISVM1 is not linearly separable, fact that can be tested immediately (in step ISVM2(b)), the hyperplane generated at step ISVM2 is dismissed and the corresponding subset is returned to the training set.

When the iteration through steps ISVM1 and ISVM2 is completed, in the sense that no other subsets satisfying the constraint specified at step ISVM1 can be generated, the hyperplanes generated at step ISVM2 are combined so as to minimize the overall training error. In fact, this error is null
since by steps ISVM1 and ISVM2, it is true that for any given training point there is at least one hyperplane which correctly classifies this point.

**Inferring the final separating surface**

Various algorithms for collecting hyperplanes to form the final separating surface can be considered. All require an evaluation of the data set used for training with respect to the hyperplanes generated. In the following let *Current* denote the hyperplane with respect to which the data points used for training are evaluated. The final separating surface is generated as follows:

1. Initialize *Current* to the smallest slope hyperplane;

2. Scan all the data points used for training and evaluate with respect to *Current*. When a training datum is misclassified by *Current*, update *Current* to be the hyperplane closest to it (in the current implementation in the \( x_1, x_2 \) space the distance along \( x_2 \) is considered).

3. Repeat Steps (1) and (2) until all data points used for training are exhausted. The collection of selected hyperplanes forms the final separating surface.

### 5.2 Simulation Results

Simulations of the Iterative SVM algorithm, implemented using the Symbolic Toolbox of MATLAB, were carried out for linearly separable data as well as for data sets where the two classes were not linearly separable. All the results shown in the tables below are obtained for 100 runs of the algorithm for the corresponding combination of distance threshold and subset size.
5.2.1 Linearly Separable Case

The data sets used for this simulation are shown in Figure 5.2. Figure 5.3(a) shows the hyperplanes obtained when subsets of size eight were generated for training with the distance threshold 11. Five hyperplanes were generated, of which two are retained in the final separating surface as shown in Figure 5.3(b). The classification accuracy obtained for the test data is 96.5%. Other results, for the cases when the distance threshold is set to 11 and 12 for the subsets of size six and eight points are shown in Table 5.1.

As it can be seen from Table 5.1 the separating surfaces obtained by the algorithm showed excellent performance (good recognition rate for the test data), maximum of 100% and over 95% in average. Even the minimum performance achieved 93% recognition rate when the distance threshold was set to 12 for the subsets of size eight. It can be observed that when a larger number of training data was used (which means that a larger threshold was selected), a better performance was obtained. The final separating surfaces used well under 50% of all the hyperplanes generated.

![Figure 5.2: Two classes separated by a linear surface](image)

Figure 5.2: Two classes separated by a linear surface
Figure 5.3: An example of the simulation results for the linearly separable data (Figure 5.2) with subset size 8 and distance threshold 11, 96.5% of classification accuracy for the test data

<table>
<thead>
<tr>
<th>distance threshold</th>
<th>subset size</th>
<th>average # of training points</th>
<th>average # of hyperplanes generated</th>
<th>average # of hyperplanes retained</th>
<th>recognition rate (%) max min average</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>6</td>
<td>70.02/100</td>
<td>11.67</td>
<td>4.69</td>
<td>99.00 85.00 95.09</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>67.04/100</td>
<td>8.38</td>
<td>3.83</td>
<td>100 76.00 95.32</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>98.70/100</td>
<td>16.45</td>
<td>5.57</td>
<td>99.50 92.50 97.24</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>98.64/100</td>
<td>12.33</td>
<td>5.19</td>
<td>99.50 93.00 97.02</td>
</tr>
</tbody>
</table>

Table 5.1: Simulation results for the linearly separable data in Figure 5.2

5.2.2 Non-Linear Case 1

The algorithm is applied to a two-dimensional data set in which two classes are separated by a degree three polynomial curve, \( y = x^3 - 3x^2 - x + 8 \), as shown in Figure 5.4. Figure 5.5(a) shows the hyperplanes obtained for one run of the algorithm with subsets of size eight and distance threshold 7.2. A total of 13 hyperplanes are generated, seven of which are used for the final surface. The resulting piecewise linear separating surface yields 93.5% of classification accuracy over the test data as shown in Figure 5.5(b). It can be seen that the final separating surface approximates in fact the polynomial curve of the actual separating surface.

The results for the simulations with the distance threshold 7.2 and 7.5, each for the subset size
of six and eight are shown in Table 5.2.

Although the overall performance is worse than the linearly separable case, it still accomplished over 91% average classification accuracy for the test data. As in the linearly separable case, setting a larger threshold improved the performance. About 50% to 60% of the hyperplanes generated at ISVM2 were retained to form the final separating surface. This corresponds to the fact that more hyperplanes are necessary to approximate the polynomial curve than to approximate the linear class boundary.
5.2.3 Non-Linear Case 2

The optimal separating surface for the data set used in this simulation is a non-linear, more tortuous curve than the polynomial curve in simulation 5.2.2. The two classes are separated by a sigmoidal curve \( y = \sin \pi x \) as shown in Figure 5.6. Table 5.3 shows all the experimental results with the distance threshold 3.2 and 3.4, the subset of size six and eight points respectively.
Figure 5.7: An example of the simulation results for the sigmoid data (Figure 5.6) with subset size 8 and distance threshold 3.4, 83.0% of classification accuracy for the test data.

An example of this simulation is shown in Figure 5.7 where the subset size is set to eight and the distance threshold is 3.4. In training, thirteen hyperplanes are generated and nine of them construct the final separating surface (Figure 5.7(a)). The resulting surface performs 83.0% correctly over the test data (Figure 5.7(b)).

The average recognition rate for the test data was below 80% for the distance threshold of 3.2 and about 82% for the distance threshold of 3.4, which are worse than those of the previous two simulations, though in individual runs it also achieved the maximum of around 90% of classification accuracy. About 60% to 70% of hyperplanes generated were retained for the final separating surface. The increase in the number of hyperplanes retained for the final separating surface over the polynomial case can be attributed to the fact that the linear approximation of the sigmoid curve in this simulation is more difficult than the approximation of the polynomial curve.
<table>
<thead>
<tr>
<th>distance threshold</th>
<th>subset size</th>
<th>average # of training points</th>
<th>average # of hyperplanes generated</th>
<th>average # of hyperplanes retained</th>
<th>recognition rate (%)</th>
<th>max</th>
<th>min</th>
<th>average</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>6</td>
<td>79.02/100</td>
<td>13.17</td>
<td>6.07</td>
<td>88.50</td>
<td>52.00</td>
<td>79.65</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>65.44/100</td>
<td>8.18</td>
<td>5.83</td>
<td>90.00</td>
<td>52.00</td>
<td>76.56</td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>6</td>
<td>94.44/100</td>
<td>15.74</td>
<td>9.14</td>
<td>88.50</td>
<td>66.50</td>
<td>81.87</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>95.52/100</td>
<td>11.94</td>
<td>8.30</td>
<td>90.50</td>
<td>70.50</td>
<td>82.23</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3: Simulation results for the linear approximation of the sigmoidal separating surface in Figure 5.6

5.3 Computational Aspects

Step BA1 is the most expensive computationally: for a subset of size $m$, this step calls for solving a system of $m + 1$ equations (in $a_i, i = 1, \ldots, m$) $m$ of which are quadratic. Therefore, the number of solution sets generated in the step ISVM2 is exponential in $m$ ($2^m$ sets of solutions), and this puts a limit on the size of the subsets sampled for training. Further complexity of the training is determined by the number of hyperplanes generated and which must be scanned at the step ISVM4. This number is controlled by the distance threshold selected. Based on the above observation, it can be said that the training is computationally efficient when the subset size is small enough to be computed at BA1 and also a small distance threshold is selected in order to force generation of fewer hyperplanes.

From the simulation results (Table 5.1 through 5.3), it can be seen that when the number of hyperplanes generated is large, which means, large number of points are used for training, a high recognition rate for test data (good performance) can be expected in average. This leads to a tradeoff between performance and efficiency: for using as many training data points as possible, a larger distance threshold is needed; however, in this case the efficiency deteriorates. Even though the average performance is worse, setting a threshold small potentially leads to high classification accuracy together with high efficiency in training. In Figure 5.8 and 5.9, 100 runs are plotted...
according to the number of hyperplanes generated in training and the corresponding recognition rate for the test data to compare the cases of small versus large threshold values. For example, Figure 5.9 shows that in simulation 5.2.3, when the training is done with the threshold set to 3.2 and the number of hyperplanes generated is around 8 to 10, can be said to be the best in both performance and efficiency.

![Graph](image1)

Figure 5.8: The number of hyperplanes generated in training (x-axis) and the corresponding recognition rate for the test data (y-axis) with subset of size six in simulation 5.2.3 (100 runs)

![Graph](image2)

Figure 5.9: The number of hyperplanes generated in training (x-axis) and the corresponding recognition rate for the test data (y-axis) with subset of size eight in simulation 5.2.3 (100 runs)
5.4 Modeling versus Generalization Power

The algorithm for constructing the final separating surface (both for the case of linearly separable data and linearly non-separable data) ensures zero classification error for the training data. In the case of linearly separable data, where the performance on the test data exceeds 90%, this is acceptable. However, for the linearly non-separable data sets, the behavior of the algorithm suggests that overfitting takes place. Overfitting refers to the situation where a learning algorithm performs very well (zero error) for the training data and significantly worse on the test data (that is, the algorithm has poor generalization ability). This occurs when the system is trained too precisely for the training data and/or when the two classes are not adequately represented in the training data (unbalanced training data set).

For the non-linear case the simulation results show acceptable performance of the algorithm ISVM for test data even under no error for the training data. Alternatively, the step ISVM₄ may be modified so as to allow errors of classification of the training data, in which case an improvement in the classification of the test data is expected. Example 5.4.1 below illustrates results for this modification when in the step ISVM₄ one classification error is accepted for each selected hyperplane.

Example 5.4.1 Step (2) of the algorithm for updating the Current hyperplane described in section 5.1 is modified as follows:

(2') Scan all the data points used for training and evaluate with respect to Current. As soon as more than one training datum is misclassified by Current, update Current to be the hyperplane closest to it (in the current implementation, in the \((x_1, x_2)\) space the distance along \(x_2\) is considered).

Table 5.4 shows the results when the modified algorithm was applied to the linear approximation
of the cubic data set from section 5.2.2. The distance thresholds 7.2 and 7.5 were used for the
original and modified algorithm which were run on the same data sets. At first glance, the average
recognition rate shown seems to indicate a slight decrease in the performance of the modified
algorithm. However, when comparing the two algorithms, the number of the hyperplanes selected
to form the final separating surface must be taken into account as well. By calculating an average
recognition rate per hyperplane the modified algorithm registers a slight increase when compared
to the original algorithm: for the distance threshold 7.2 the modified algorithm has a 14.76%
recognition rate per hyperplane, versus that of 12.91% for the original algorithm; for the distance
threshold of 7.5, the modified algorithm has a 13.66% recognition rate per hyperplane versus 11.73%
for the original algorithm. For both distance thresholds the modified algorithm introduced an
average training error of 7%. This example illustrates the tradeoff between the modeling power,
the generalization power of the final separating surface and the number of hyperplanes that make
it. This tradeoff can be adjusted by varying the number of errors allowed for any given hyperplane.
Chapter 6

Piecewise Non-Linear Approximation

6.1 Non-Linear Transformation in the Original Data Space

The idea to be considered in this section is to represent the non-linear separating surface by pieces of simple curves (e.g. pieces of parabolas) by extending the approach in the previous sections.

In chapter 3, it is shown that the linear version SVM returns the separating hyperplane $D(x)$ for the $m$-dimensional data as:

$$D(x) = w^T x + b$$  \hspace{1cm} (6.1)

Now, suppose for simplicity, the input data is two-dimensional $(x_{i1}, x_{i2}, y_i)(i = 1, \ldots, M)$. Then, (6.1) can be rewritten as

$$D(x) = w_1 x_1 + w_2 x_2 + b$$  \hspace{1cm} (6.2)

It can be observed that (6.2) can be turned into a non-linear surface by performing a non-linear
transformation, \( f \) on \( x_1 \):

\[
x = (x_1, x_2) \to X = (f(x_1), x_2) = (X_1, x_2)
\]  

(6.3)

in which case (6.2) becomes a non-linear function for \( x_1 \)

\[
D(x) = w_1 f(x_1) + w_2 x_2 + b
\]  

(6.4)

However, (6.4) can also be seen as a linear surface in the space \((X_1, x_2)\).

\[
D(X) = w_1 X_1 + w_2 x_2 + b
\]  

(6.5)

Therefore, the linear separating surface \( D(X) \) in \( X = (X_1, x_2) \) space corresponds to the non-linear separating surface \( D(x) \) given by (6.4) in \((x_1, x_2)\), the original space.

The above discussion suggests that a non-linear separating surface can be obtained by SVM by performing a Non-Linear Transformation ("bending") of one (or more) axes of its input space.

Figure 6.1 lists the steps of the procedure \( NLT \) in which non-linear transformation of an attribute can be used in conjunction with the SVM algorithm to infer a non-linear separating surface.

| \( NLT1 \) | Select an attribute of input data and a non-linear function \( f \). |
| \( NLT2 \) | Transform the selected attribute by \( f \). |
| \( NLT3 \) | Find a separating hyperplane for the transformed data by linear version SVM. |
| \( NLT4 \) | Transform the hyperplane back into the original data space. |

Figure 6.1: The Non-Linear Transformation for obtaining a non-linear separating surface by SVM
The procedure $NLT$ is illustrated in a simple example for the two-dimensional data and the non-linear transformation of $f(t) = t^2$ is described below.

Example

The two-dimensional data shown in Figure 6.2(a) is linearly non-separable.

![Figure 6.2](image)

**Figure 6.2:** Simple example of the Non-Linear Transformation

First, according to steps $NLT1$ and $NLT2$, the attribute $x_1$ is mapped into by $X_1 = f(x_1) = x_1^2$ which yields the new attribute space:

$$X = (X_1, x_2) = (x_1^2, x_2)$$  \hspace{1cm} (6.6)

Next, according to the step $NLT3$ the linear version SVM is applied in the new space $X = (X_1, x_2)$. The obtained hyperplane $D(X)$ shown in Figure 6.2(b) and equation (6.7).

$$D(X) = w_1 X_1 + w_2 x_2 + b$$  \hspace{1cm} (6.7)

Finally, according to the step $NLT4$, transforming (6.7) back into the original data space $(x_1, x_2)$,
leads the quadratic (in $x_1$) separating surface shown in Figure 6.2(c) and equation (6.8).

$$D(x) = w_1 x_1^2 + w_2 x_2 + b$$ (6.8)

To notice here is that different from the data mapping with a kernel described in section 3.4 which increases dimensionality, the proposed non-linear transformation above does not increase the dimensionality of the input space, which means it does not affect the size of the matrix computed by SVM. Although it requires careful selection of the attribute (axis) of the input data to be transformed and of the actual non-linear transformation $f$, it suggests that the non-linear separating surface can be obtained by a linear version of SVM without adding extra computational complexity.

### 6.2 Piecewise Non-Linear Approximation

Combining the non-linear transformation and the iterative SVM algorithm described in section 5.1 suggests that it is possible to compose an approximation of the real non-linear arbitrary separating surface by a combination of piecewise non-linear curves of selected $f$. The iterative version of NLT algorithm, INLA, which achieves this is shown in Figure 6.3.

First, in the step INLA1 the non-linear transformation of the selected attribute(s) of the input data is performed by the selected transformation $f$ and a new data space is generated. It is desired that $f$ is selected so as to induce the high linear separability in the new data space to expect good approximation by the application of Iterative SVM algorithm in step INLA2. If $f$ corresponds to the actual optimal separating surface best results of the order shown for the separable case of section 5.2.1 are expected. In general, however, the optimal separating surface is not known and this is why it is needed to be approximated.

At step INLA2, linear hyperplanes are generated by applying the first three steps of the Iterative
**INLA 1.** Produce the transformed input space by mapping attribute(s) of input data by a non-linear transformation $f$.

**INLA 2.** Apply ISVM1 - ISVM3 of the Iterative SVM algorithm in the transformed space and obtain linear hyperplanes $h_j$, $j = 1, \ldots, n$, ($n$: the number of hyperplanes generated).

**INLA 3.** Transform $h_j$ back into the original data space.
The linear hyperplanes $h_j$ appear as non-linear surfaces $g_j$.

**INLA 4.** Infer the overall separating surface by combining $g_j$.

Figure 6.3: The Iterative Non-Linear Approximation Algorithm

SVM algorithm. To avoid introducing classification errors in the original data space, the hyperplanes are not combined in the transformed space. Instead, at step INLA3, they are mapped back into the original data space and at step INLA4 they are combined in the same way as described in section 5.1.

### 6.3 Simulation Results

Several experiments of the Non-Linear Approximation Algorithm are conducted for the non-linear data sets used for the simulations in section 5.2.2 and 5.2.3. The algorithm was run 100 times for each combination of distance threshold and subset size selected.

#### 6.3.1 Quadratic Approximation of Non-linear Separating Surfaces

The non-linear transformation $f(t) = t^2$ is applied to the $x_1$ coordinate. The idea behind this choice for $f$ is that of approximating the separating surface by quadratic pieces, in a way, similar to the idea behind the Simpson rule for approximate integration.
<table>
<thead>
<tr>
<th>distance threshold</th>
<th>subset size</th>
<th>average # of training points</th>
<th>average # of hyperplanes generated</th>
<th>average # of hyperplanes retained</th>
<th>recognition rate (%) max</th>
<th>min</th>
<th>average</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2</td>
<td>6</td>
<td>77.52/100</td>
<td>12.92</td>
<td>6.99</td>
<td>97.50</td>
<td>70.50</td>
<td>89.99</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>53.36/100</td>
<td>6.67</td>
<td>4.42</td>
<td>97.00</td>
<td>77.00</td>
<td>88.25</td>
</tr>
<tr>
<td>7.5</td>
<td>6</td>
<td>95.22/100</td>
<td>15.87</td>
<td>8.25</td>
<td>97.00</td>
<td>75.00</td>
<td>91.05</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>93.84/100</td>
<td>11.73</td>
<td>7.18</td>
<td>97.00</td>
<td>84.00</td>
<td>91.48</td>
</tr>
</tbody>
</table>

Table 6.1: Simulation results for the quadratic approximation of the cubic separating surface in Figure 5.4.

### 6.3.2 Quadratic Approximation of the Cubic Separating Surface

In this simulation, the cubic separating surface in Figure 5.4 is approximated with the $x^2$ curves by transforming the $x_1$ coordinate by $f(t) = t^2/(b - a)$, where $[a, b]$ denotes the domain of values for $x_1$.

Figure 6.4(a) shows the linear hyperplanes obtained in the transformed data space (total of thirteen hyperplanes) when subset size is set to eight and the distance threshold is set to 7.5. Figure 6.4(b) shows the non-linear surfaces obtained by mapping these separating hyperplanes back into the original data space and the final separating surface combined at step INLA4. The final separating surface consists of six non-linear surfaces correctly classifies 94% of the test data (Figure 6.4(c)).

Other results for the simulations with the distance threshold set to 7.2 and 7.5, each for the subset size of six and eight are shown in Table 6.1. The average performance is slightly worse comparing to the results from the linear approximation for the same polynomial curve in section 5.2.2. Although in this simulation taking a larger threshold (7.5 versus 7.2) does not make a significant improvement of the maximum recognition rate, it does make about 6% improvement of the minimum and therefore average recognition rate.
<table>
<thead>
<tr>
<th>distance threshold</th>
<th>subset size</th>
<th>average # of training points</th>
<th>average # of hyperplanes generated</th>
<th>average # of hyperplanes retained</th>
<th>recognition rate (%) max</th>
<th>min</th>
<th>average</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.8</td>
<td>6</td>
<td>51.60/100</td>
<td>8.60</td>
<td>5.53</td>
<td>89.00</td>
<td>37.50</td>
<td>77.02</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>55.04/100</td>
<td>6.88</td>
<td>5.07</td>
<td>87.00</td>
<td>63.50</td>
<td>74.41</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>99.87/100</td>
<td>16.79</td>
<td>9.64</td>
<td>89.00</td>
<td>67.50</td>
<td>83.40</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>99.79/100</td>
<td>12.59</td>
<td>8.57</td>
<td>89.00</td>
<td>72.50</td>
<td>82.62</td>
</tr>
</tbody>
</table>

Table 6.2: Simulation results for the quadratic approximation of the sigmoid separating surface in Figure 5.6.

6.3.3 Quadratic Approximation of the Sigmoid Separating Surface

The Non-Linear Approximation Algorithm is applied to the sigmoid data shown in Figure 5.6. The linear hyperplanes generated in the transformed data space are shown in Figure 6.5(a). Seventeen hyperplanes are generated with the subsets of size six and the distance threshold 9. In Figure 6.5(b), these hyperplanes are transformed into the original data space and appear as non-linear surfaces. The final separating surface performs 84.5% on the test data (Figure 6.5(c)). As it can be seen, the resulting separating surface is comparable to that obtained from the linear approximation suggesting that the selection of \( t^2 \) for \( f \) does not improve over the initial data space.

Table 6.2 shows all the simulation results with the distance threshold 8.8 and 9, the subset of size six and eight points, respectively. A slightly better recognition rate was achieved over the linear approximation with distance threshold 3.4. However, it cannot be said that this improvement is because of the non-linear transformation, since the number of points used for the training is greater than that of linear approximation.

6.3.4 Cubic Approximation of Non-linear Separating Surfaces

This section describes experiments of approximating the cubic separating surface for the data sets in Figure 5.4 by \( t^3/(b - a)^2 \), where \([a, b]\) denotes the domain of \( x_1 \). Figure 6.6 shows the results for one run of this approximation: Figure 6.6(a) shows the transformed data space and all the...
<table>
<thead>
<tr>
<th>distance threshold</th>
<th>subset size</th>
<th>average # of training points</th>
<th>average # of hyperplanes generated</th>
<th>average # of hyperplanes retained</th>
<th>recognition rate (%) max</th>
<th>recognition rate (%) min</th>
<th>recognition rate (%) average</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2</td>
<td>6</td>
<td>78.96/100</td>
<td>13.16</td>
<td>7.57</td>
<td>96.00</td>
<td>67.50</td>
<td>88.23</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>71.44/100</td>
<td>8.93</td>
<td>5.83</td>
<td>96.50</td>
<td>65.00</td>
<td>88.74</td>
</tr>
<tr>
<td>7.5</td>
<td>6</td>
<td>97.14/100</td>
<td>16.19</td>
<td>8.84</td>
<td>96.50</td>
<td>80.50</td>
<td>90.74</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>94.24/100</td>
<td>11.78</td>
<td>7.68</td>
<td>96.50</td>
<td>77.50</td>
<td>90.68</td>
</tr>
</tbody>
</table>

Table 6.3: Simulation results for the cubic approximation of the cubic separating surface in Figure 5.4.

linear hyperplanes generated for the transformed data. Thirteen hyperplanes are generated by the algorithm when the subset size is set to eight and the distance threshold is 7.5. In Figure 6.6(b), the linear hyperplanes in the transformed space are mapped back to the original data space where they appear as polynomial curves of degree three. The final separating surface recognizes 94% correctly over the test data (Figure 6.6(c)).

Other results are shown in Table 6.3. The distance threshold is set to 7.2 and 7.5 for the subset of size six and eight points respectively. The overall performance is worse than the quadratic approximation with \( t^2 \) curve in the previous section and this is to be expected. Over 50% to 70% of the generated surfaces were used to construct the final separating surface.

6.4 Conclusion

Comparing the simulation results in the previous section and those in chapter 5, even though a slight improvement was made for the sigmoid case, the approximation with non-linear curves does not seem to improve its performance over the piecewise linear approximation. Especially when the resulting separating surfaces for the sigmoid data from the linear approximation (Figure 5.7) and non-linear approximation (Figure 6.5) are compared, it appears that the non-linear approximation may not utilize the generated curves in an optimal way (its results being comparable to those of the linear approximation). The example in Figure 6.7 shows the result under an optimal selection
for the transformation $f$: identical to the actual separating surface. This choice leads to very good performance. However, in general, the function for the optimal separating surface is not known.

Further research towards a theory of selection an appropriate non-linear transformation $f$ for the data is needed.

On the other hand, the results of the current study show that the linear classifier SVM can produce a non-linear separating surface by introducing the non-linear transformation which does not increase the data dimensionality. Instead of employing the same procedures as the piecewise linear approximation, more appropriate algorithm to decide the final separating surface with this method should be considered.
(a) Linear hyperplanes generated in the transformed data space.

(b) Non-linear hyperplanes in the original data space.

(c) Final separating surface on test data (94% classification accuracy).

Figure 6.4: Quadratic approximation of the cubic separating surface in Figure 5.4 with subset size 8 and distance threshold 7.5.
(a) Linear hyperplanes generated in the transformed data space

(b) Non-linear hyperplanes in the original data space

(c) Final separating surface on test data (84.5% classification accuracy).

Figure 6.5: Quadratic approximation of the sigmoidal separating surface in Figure 5.6 with subset size 6 and distance threshold 9.
(a) Linear hyperplanes generated in the transformed data space.

(b) Non-linear hyperplanes in the original data space.

(c) Final separating surface on test data (94% of classification accuracy).

Figure 6.6: Cubic approximation of the cubic separating surface in Figure 5.4 with subset size 8 and distance threshold 7.5.
Figure 6.7: Optimal curve approximation of the cubic separating surface in Figure 5.4 with subset size 6 and distance threshold 10.
Chapter 7

Conclusion and Future Work

7.1 Conclusion

The Support Vector Machine approach remains a lively area of research, with many issues continuing to arise regarding both its theoretical aspects and practical implementations. The work presented in this thesis used only the linear (hard margin) version SVM, without mapping a higher dimensional feature space. Several experiments were carried out for linearly separable data as well as non-linear data whose optimal separating surface is a polynomial curve and a sigmoidal curve. The results showed good approximations of the true separating surface. More precisely, the work presented in this thesis focused on two issues:

The first issue regards obtaining an approximate separating surface for a training set as a collection of exact solutions for the linear SVM for selected subsets of it. This approach is different from approaches such as [12] but somewhat similar to approaches such as [11]. Still, unlike [11] the current approach does not make use of a higher dimensional feature space and therefore of kernel functions. This means that for linearly non-separable data sets the current approach results in a piecewise linear separating surface. Results on selected nonlinear classes of actual separating
surface (cubic and sigmoidal) suggest a good performance of this approach.

The second issue considered in this thesis is that of extending the piecewise linear approximation of the separating surface to piecewise nonlinear approximation of the actual separating surface when the nonlinearity of the pieces of the final surface is prescribed in advance (as shown in chapter 6 where experiments of approximating cubic and sigmoidal actual separating surface by quadratic and cubic piecewise surfaces were carried out). All results must be compared to the performance of the algorithm on linearly separable data, as shown in Table 5.1 which achieves around 97% classification accuracy of the test data and in this light even the worst average performance is acceptable.

7.2 Future Work

The experiments conducted in this work showed encouraging initial results with the proposed method. However, several issues arise in connection with the proposed approach. Elucidation of these issues is expected to further clarify the contribution of this approach and to improve upon the results presented. In addition, challenges for more difficult problems where the proposed approach can be applied are interesting to examine. This section describes the issues that need further attention and possible directions of future research in connection with these issues.

7.2.1 Selection of Training Subsets

In the current implementation, the selection of the training subsets is done under a mild distance threshold constraint aimed at ensuring that the data points selected were from around the actual class boundary. Other constraints can be used in conjunction with the distance threshold. For example, the slopes of the intermediate hyperplanes can be used to further constrain selection of the subsets generated subsequently. This would be more sensitive to variations in the slope of the
actual class boundary.

7.2.2 Use of Other Kernels

The hard margin SVM used in this work corresponds to the hard linear kernel. The initial motivation for the proposed approach is to avoid (possibly unintuitive) implicit mappings into a higher dimensional space corresponding to different selections of kernels. However, the method can be applied when other kernels are selected: for example, to investigate the extent to which arbitrary linearly non-separable data can be well separated by piecewise surfaces corresponding to polynomial kernels.

7.2.3 Non-Linear Transformation of Attribute Axes

As an alternative to the use of kernels, the piecewise approximation was extended to nonlinear approximations. A more theoretical direction of study could be that of characterizing the (non-linearity of) class boundaries according to the nonlinearity of the class of functions used to approximate it. For example, find those class boundaries which are best (or within a specified error) approximated by pieces of quadratic surfaces.

7.2.4 Inference of the Final Separating Surface

In the method illustrated in this work, the final separating surface is formed so as not to make any error over the training points. When training errors are allowed (Example 5.4.1), an improvement in the efficiency (though not necessarily in the overall accuracy) of the algorithm was observed.

Furthermore, currently, when the transition to a new hyperplane is needed, the one which is closest to the current hyperplane is selected so as to avoid generating a rough surface. Alternatives for the selection of the hyperplanes making up the final separating surface should be considered. For
example, in informal initial experiments, it was observed that for the non-linear approximation, the intersection points of the non-linear surfaces generated may lead to a smoother and more performant final separating surface.

7.2.5 Approximations of Difficult Class Boundaries

The proposed method has exhibited the validity for some cases with non-linear class boundary. However in all the examples considered, the actual separating surface corresponded to a one-to-one function. The next challenge to consider in this direction would be to extend the approach for the more difficult class boundaries, such as circular or closed polyhedral shapes as shown in Figure 2.1(d)(e). For such problems, the current method (step ISVM4) would need to be modified so as to allow the selection of more than one hyperplane.
Bibliography


