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 on going 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 take an appropriate action for the situation even when this situation is new. Imagine, you have just moved to a new town. You might want to walk around to know the surroundings better. For the first time, you might get lost, or you might have to take a long walk to get to the nearest grocery store. Then, what happens to you after a week later? You know 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 its 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 [2] [1]. 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

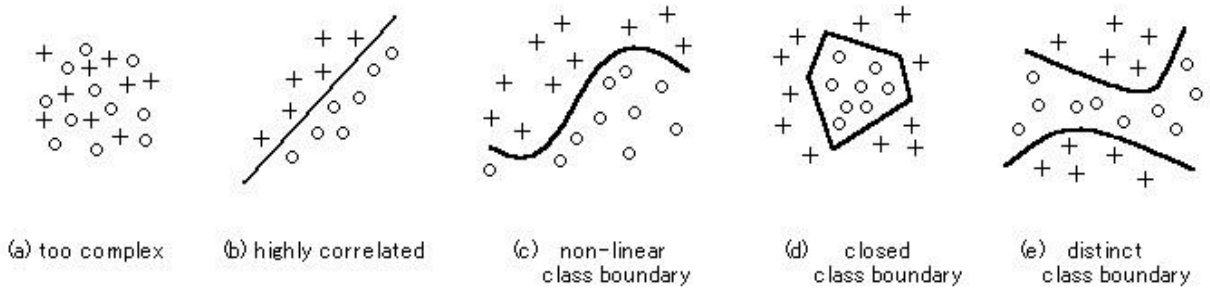


Figure 2.1: Difficult cases in classification

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.

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]. 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, infinite number of separating surfaces which correctly classify those data may 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

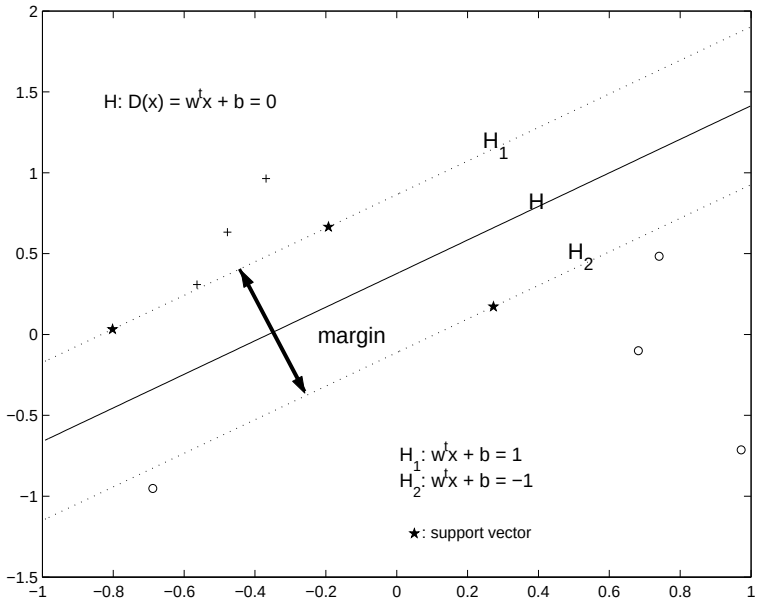


Figure 3.1: support vectors, hyperplanes, margin

fact can be utilized to reduce the number of training data actually used for the training and improve the training effect.

There are two types of linear version of SVM, hard margin SVM and soft margin SVM. 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 "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 "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 $class_1$ if $y_i = +1$ or $class_2$ if $y_i = -1$, the separating hyperplane, $D(x)$, of the two classes is given by

$$D(x) = w^t x + b \tag{3.1}$$

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

$$w^t x_i + b \geq +1, \quad for \;\; y_i = +1 \tag{3.2}$$

$$w^t x_i + b \leq -1, \quad for \;\; y_i = -1 \tag{3.3}$$

which can be rewritten as

$$y_i(w^t x_i + b) \geq 1, \quad i = 1, \ldots, M \tag{3.4}$$

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

$$Minimize : \frac{1}{2}||w||^2 \tag{3.5}$$

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 α_i $(i = 1, \ldots, M)$, the constrained problem given by (3.5) and (3.4) is converted into the unconstrained problem:

$$Q(w, b, \alpha) = \frac{1}{2}w^t w - \sum_{i=1}^{M} \alpha_i \{y_i(w^t x_i + b) - 1\} \tag{3.6}$$

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

$$\frac{\delta Q(w, b, \alpha)}{\delta b} = 0 \tag{3.7}$$

$$\frac{\delta Q(w,b,\alpha)}{\delta w} = 0 \tag{3.8}$$

Using (3.6), (3.7) becomes

$$\sum_{i=1}^{M} \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad for \;\; i = 1, \ldots, M \tag{3.9}$$

and (3.8) becomes

$$w = \sum_{i=1}^{M} \alpha_i y_i x_i, \quad \alpha_i \geq 0, \quad for \;\; i = 1, \ldots, M \tag{3.10}$$

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

$$Maximize: \;\; Q(\alpha) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j x_i^t x_j \tag{3.11}$$

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

$$\sum_{i=1}^{M} \alpha_i y_i = 0 \tag{3.12}$$

$$\alpha_i \geq 0, \quad for \;\; i = 1, \ldots, M \tag{3.13}$$

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

$$\alpha_i \{ y_i (w^t x_i + b) - 1 \} = 0 \tag{3.14}$$

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

$$D(x) = \sum_{i=1}^{M} \alpha_i y_i x_i^t x + b \tag{3.15}$$

where

$$b = y_i - w^t x_i, \quad for \ \ all \ \ \alpha_i \neq 0 \tag{3.16}$$

Classification of a data point is done according to the sign of the hyperplane evaluated at that point, that is, x is classified into $class_1$ if $D(x) > 0$ and into $class_2$ 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 "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,

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

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

$$Minimize : \frac{1}{2}||w||^2 + C\sum_{i=1}^{M} \xi_i \tag{3.18}$$

subject to the constraints (3.17) and

$$\xi_i \geq 0, \quad for \;\; i = 1, \ldots, M \tag{3.19}$$

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, α and β for the constraints (3.17) and (3.19) are needed. Then, (3.18), (3.17) and (3.19) becomes,

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

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

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

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

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

From (3.20), these become, respectively,

$$\sum_{i=1}^{M} \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad for \;\; i = 1, \ldots, M \tag{3.24}$$

$$w = \sum_{i=1}^{M} \alpha_i y_i x_i, \quad \alpha_i \geq 0, \quad for \;\; i = 1, \ldots, M \tag{3.25}$$

$$\alpha_i + \beta_i = C, \quad \alpha_i, \beta_i \geq 0, \quad for \;\; i = 1, \ldots, M \tag{3.26}$$

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

$$Maximize: \; Q(\alpha) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j x_i^t x_j \tag{3.27}$$

subject to

$$\sum_{i=1}^{M} \alpha_i y_i = 0 \tag{3.28}$$

$$0 \leq \alpha_i \leq C, \quad for \;\; i = 1, \ldots, M \tag{3.29}$$

Also, the optimal solution must satisfy the KKT conditions:

$$\alpha_i \{ y_i (w^t x_i + b) - 1 + \xi_i \} = 0 \tag{3.30}$$

$$\beta_i \xi_i = (C - \alpha_i) \xi_i = 0 \tag{3.31}$$

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

3.4 Kernel Functions "non-linear case"

When the optimal separating surface for the data is non-linear, SVM first map the input data by ϕ 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).

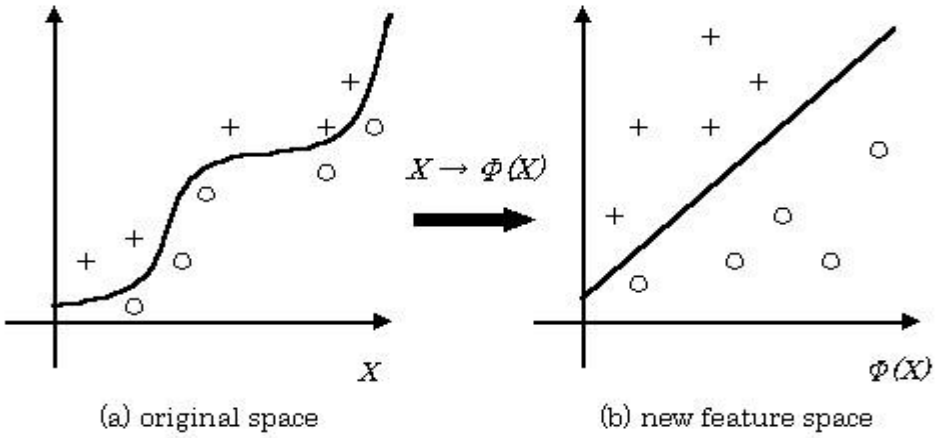


Figure 3.2: Data mapping by a kernel

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

$$K(x_i, x_j) = \phi(x_i)^t \phi(x_j) \tag{3.32}$$

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. The discussion about Mercer's theorem is beyond the scope of this research.

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

$$x_i^t x_j \leftarrow K(x_i, x_j) = \phi(x_i)^t \phi(x_j) \tag{3.33}$$

The optimization problem to be solved becomes

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

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

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

where

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

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 Problems

Training the SVM, equivalent to solve the convex optimization problem defined in 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 in their memory and time requirements for a large training set. Sequential Minimal Optimization (SMO) and SVM^{light} are two major implementation techniques currently used in SVM studies. 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. 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 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 [8]. Widely used implementation techniques in current SVM research, Sequential Minimal Optimization (SMO) [9] [6] and SVM^{light} [10] are based on this approach. They differ in their ways of chunking into small QP tasks. The approach considered in the following sections also exploits Osuna's idea, 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]. Accordingly, the steps taken to find the optimal solution are described in the *Basic Algorithm* of Figure 4.1.

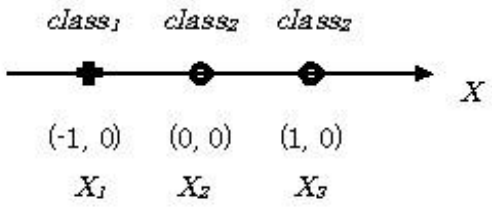
BA1.	Solve for $\alpha_i, i = 1, \ldots, M$ and b, the KKT conditions (3.14) and (3.12).
BA2.	Eliminate from the solutions obtained at *BA1*, those which do not satisfy (3.13).
BA3.	Select from the remaining candidates those which maximize $Q(\alpha)$.

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.

Example: 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).



X	Y	
-1	+1	class1
0	-1	class2
1	-1	class2

Figure 4.2: Training data in this example (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 *BA1*, the following four equations ((1), (2), (3) are from KKT conditions of (3.14) for these three training data, (4) is the constraint from (3.12)) are to be solved.

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

$\alpha_1, \alpha_2, \alpha_3$: *undetermined*

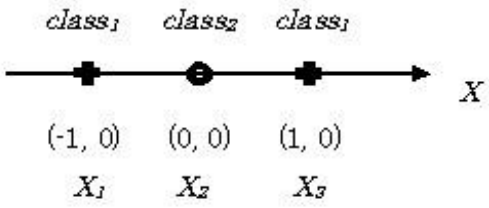
$b = -1$

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: Linearly Non-separable Case

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



X	Y	
-1	+1	$class_1$
0	-1	$class_2$
1	+1	$class_1$

Figure 4.3: Training data in this example (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 *BA1*, the following four equations ((1), (2), (3) are from KKT conditions of (3.14) for these three training data, (4) is the constraint from (3.12)) are to be solved.

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

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

$b = 1$

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

$\alpha_1, \alpha_2, \alpha_3 : undetermined$

$b = -1$

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

The solusions from (f) and (g) maximize $Q(\alpha)$ at *BA3*. The hyperplane obtained according to the solusion from (f) is given by $x_2 = 2x_1 - 1$ having $(0, 0)$ and $(1, 0)$ as support vectors. Similarly, the hyperplane obtained for the solusion from (g) is given by by $x_2 = -2x_1 - 1$ having $(-1, 0)$ and $(0, 0)$ as support vectors. However, neither of the hyperplanes separates the classes correctly.

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 are remained after the step *BA2*, and the one which maximizes the objective function $Q(\alpha)$ is selected as the optimal solution at step *BA3*. 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, all the computation at each step is done

analytically.

Chapter 5

Piecewise Linear Approximation

5.1 Iterative SVM algorithm

Considering the problems of SVM described in section 3.5, the challenge is to approximate a non-linear separating surface by using only linear version SVM, here, the hard margin SVM, without any data mapping into a higher dimensional space [11]. The resulting separating surface consists of the combination of hyperplanes 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 accomplish this.

In *ISVM1*, the random selection can be constrained by various conditions. The idea behind constraining the selection of small subsets of the training data is to ensure that those are from around the boundary between the two classes. This idea appears in other work as well [12] [13] [14]. In the current implementation, a constraint on the distance between the two classes, as represented in this subset is enforced. The reason to introduce constrained-random selection, not random selection, is to avoid generating hyperplanes which would be generated by training data points which are very far from the true support vectors.

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.
(b) If the subset is not linearly separable,
discard the hyperplane generated for it and
the subset is returned to the training set.

ISVM3. Repeat *Step1* and *Step2* until no more subsets can be generated.

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

Figure 5.1: The Iterative SVM Algorithm for an arbitrary set

If the training subset generated at step *ISVM1* is not linearly separable, which can be tested immediately, 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 *ISVM3* 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. Various altorithms for collecting hyperplanes to form the final separating surface can be considered. Here, the hyperplanes are combined as follows: First, a hyperplane which has the smallest slope is selected as the starting hyperplane. Then, data points used for training are scanned and for each its classification is evaluated with respect to the hyperplane; as soon as a training datum is wrongly classified by the current hyperplane, a

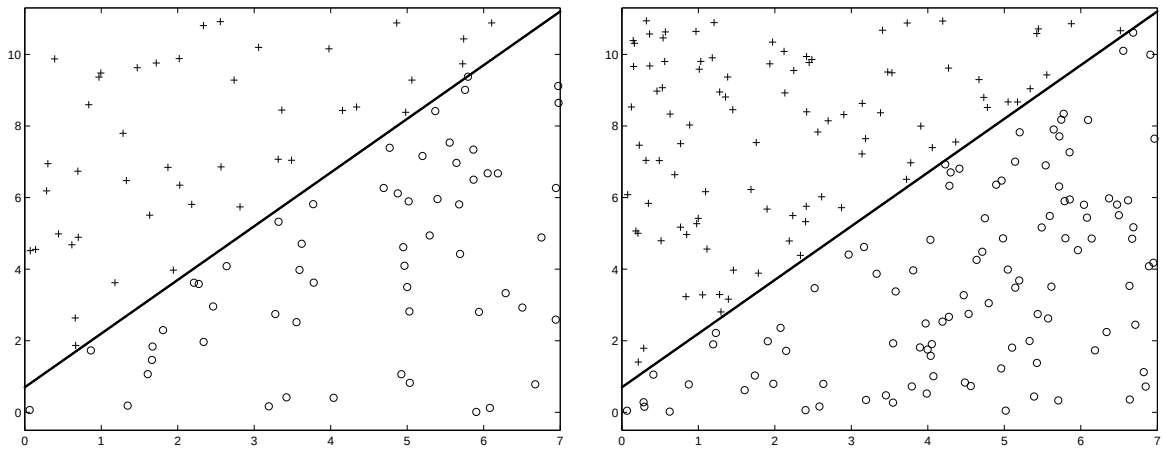
new hyperplane which correctly classifies the point and locates closest to the current hyperplane is selected. When all the training points are scanned, the final separating surface is completed.

5.2 Simulation Results

Simulations of the Iterative SVM algorithm 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 as a table in the following each section are obtained for 100 runs of the algorithm for the corresponding combination of distance threshold and subset size.

5.2.1 Linearly Separable Case

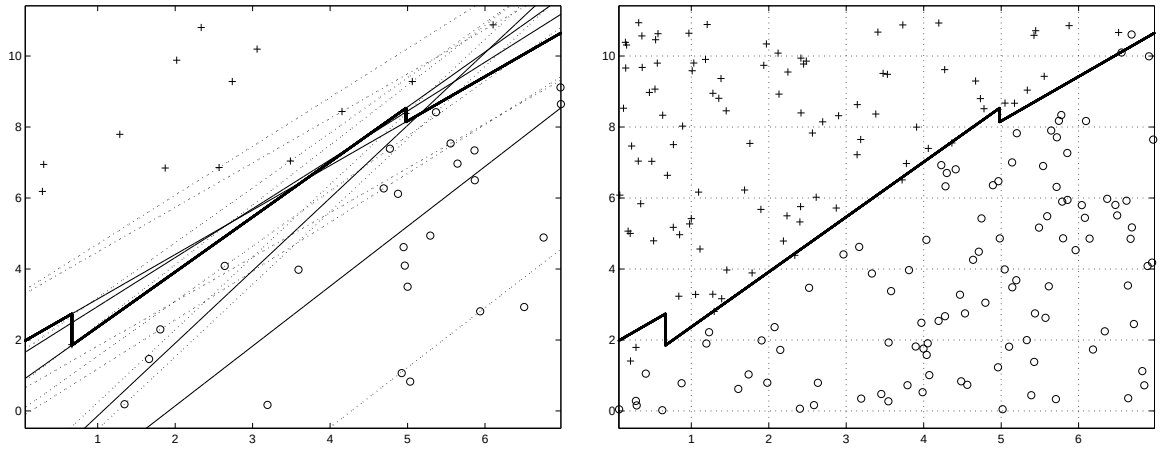
The data sets used for this simulation are shown in Figure 5.2.



(a)Training data (100 points) (b)Test data (200 points)

Figure 5.2: Linearly separable data sets and the optimal separating surface

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%.



(a)Hyperplanes generated in training (b)Final separating surface on test data

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

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

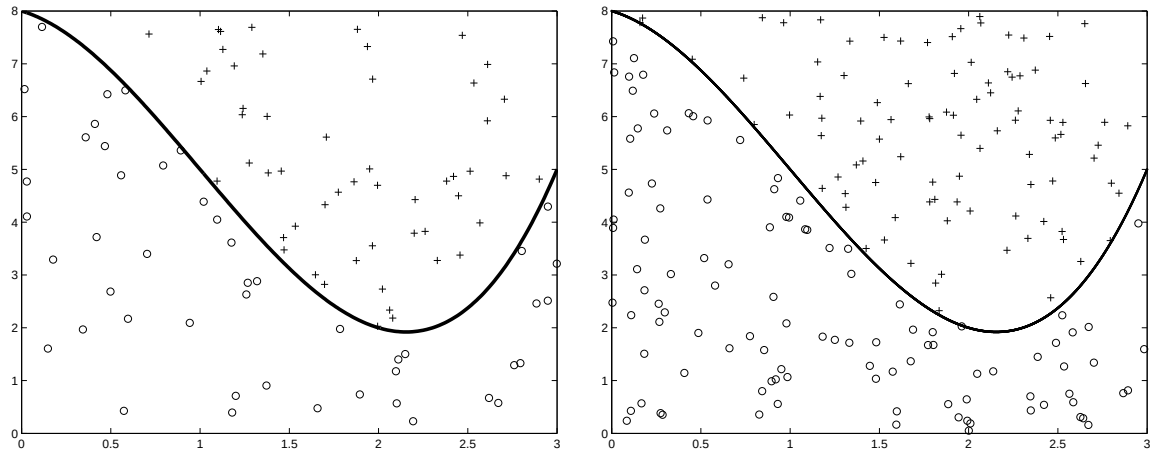
Table 5.1: Simulation results for the linearly separable data in Figure 5.2

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, 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% when the distance threshold was set to 12 for the subsets of size eight. The larger number of training data was used for the training, which means the larger threshold was taken, the better performance could be obtained. The final separating surfaces were constructed by utilizing well less than 50% of all the hyperplanes generated.

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.



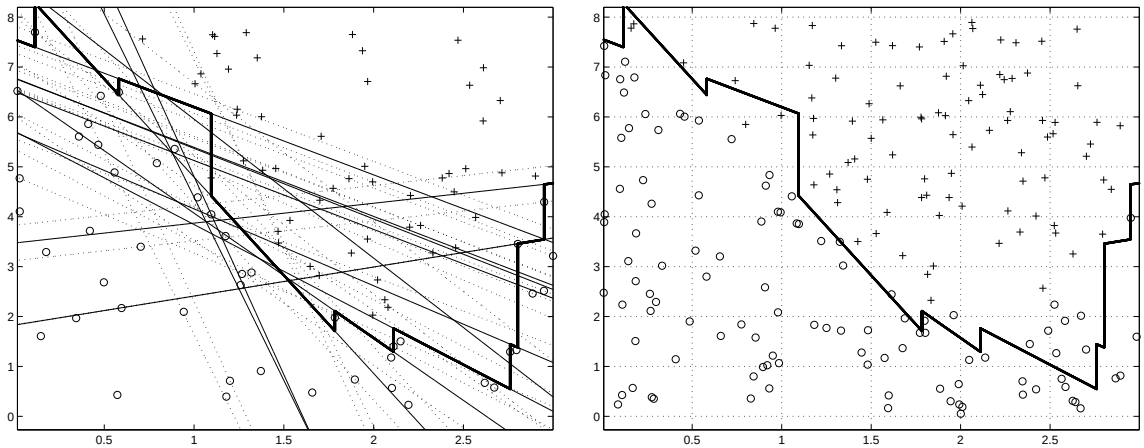
(a)Training data (100 points) (b)Test data (200 points)

Figure 5.4: Two classes separated by a polynomial curve

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 thirteen 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 points are shown in Table 5.2.

Although the overall performance is worse than the linearly separable case, it still accomplished over 91% of the average classification accuracy for the test data. Setting the larger threshold improved the performance as well as the linearly separable case. About 50% to 60% of the hyperplanes generated at *ISVM2* were retained to form the final separating surface. It can be considered that



(a)Hyperplanes generated in training (b)Final separating surface on test data

Figure 5.5: An example of the simulation results for the polynomial data in Figure 5.4 with subset size = 8 and distance threshold = 7.2, 93.5% of classification accuracy for the test data

distance threshold	subset size	average # of training points	average # of hyperplanes		recognition rate (%)		
			generated	retained	max	min	average
7.2	6	84.72/100	14.12	7.04	96.50	73.50	91.67
	8	79.04/100	9.88	5.75	96.50	79.00	91.12
7.5	6	98.16/100	16.36	7.89	97.50	72.50	92.70
	8	97.92/100	12.24	6.87	97.00	86.50	93.13

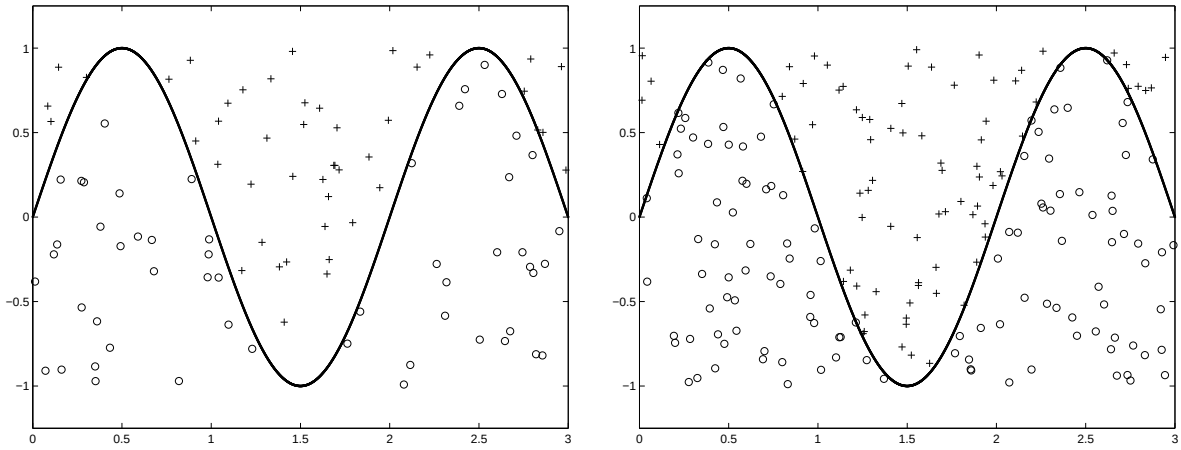
Table 5.2: Simulation results for the polynomial data in Figure 5.4

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.

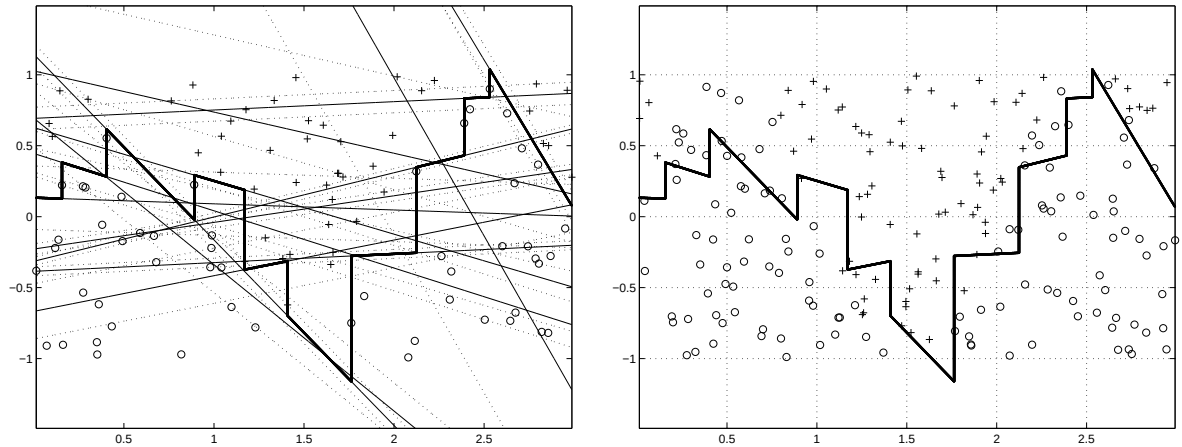
An example of this simulation is shown in Figure 5.7. Subset size is set to eight points, and the distance threshold is 3.4. In training, thirteen hyperplanes are generated and nine of them construct



(a)Training data (100 points) (b)Test data (200 points)

Figure 5.6: Two classes separated by a sigmoidal curve

the final separating surface (Figure 5.7(a)). The resulting surface performs 83.0% correctly over the test data (Figure 5.7(b)).



(a)Hyperplanes generated in training (b)Final separating surface on test data

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

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.

The average recognition rate for the test data was below 80% for the distance threshold of 3.2

distance threshold	subset size	average # of training points	average # of hyperplanes		recognition rate (%)		
			generated	retained	max	min	average
3.2	6	79.02/100	13.17	8.07	88.50	52.00	79.65
	8	65.44/100	8.18	5.83	90.00	52.00	76.56
3.4	6	94.44/100	15.74	9.14	88.50	66.50	81.87
	8	95.52/100	11.94	8.30	90.50	70.50	82.23

Table 5.3: Simulation results for the sigmoid data in Figure 5.6

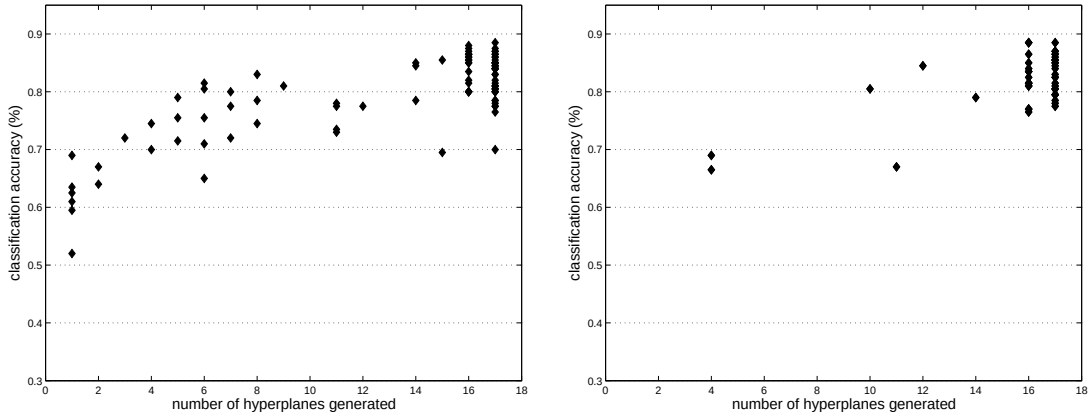
and about 82% for the distance threshold of 3.4, which are significantly worse than those of the previous two simulations, though it achieved the maximum of around 90% of classification accuracy. About 60% to 70% of hyperplanes generated were retained for the final separating surface. The increase in the number of hyperplanes retained for the final separating surface over the polynomial case can be considered because the apploximation of the sigmoid curve in this simulation is more difficult than the approximation of the polynomial curve by this method.

5.3 Computational Aspects

The approach proposed in this work was implemented in MATLAB making use of the Symbolic Tool Box for the step *BA1* which is the most expensive computationally. For the subset size m, this step calls for solving a system of $m+1$ equations, m of which are quadratic in $\alpha_i, i = 1, \ldots, m$. Therefore, the number of solution sets generated at *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*. The number of hyperplanes generated is actually equal to the number of times the step *BA1* is called by the algorithm. This number is controlled by the distance threshold selected. Based on the obove observation, it can be said that the training is computationally efficient when the subset size is set to small enough to be computed at *BA1* and also the small distance threshold

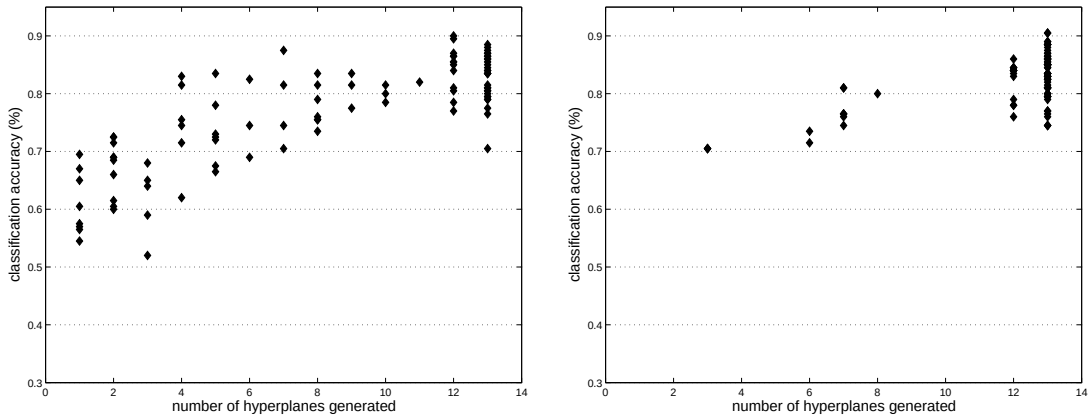
is selected in order to force the fewer number of hyperplanes to be generated.

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 data points are used for the training, high recognition rate for test data (good performance) can be expected in average. Here, a bottleneck lies between its performance and efficiency: for making use of data points as many in training, larger distance threshold should be taken, then the training efficiency would become worse. However, even though the average performance is worse, setting a threshold small has a potentiality to lead 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 with small threshold and large threshold. For example, Figure 5.9 shows that in simulation 5.2.3, when the training is done with the threshold is 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.



(a)Distance threshold = 3.2 (b)Distance threshold = 3.4

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 size = 6 in simulation 5.2.3 (100 runs)



(a)Distance threshold = 3.2 (b)Distance threshold = 3.4

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 size = 8 in simulation 5.2.3 (100 runs)

5.4 Observation

In training a system, there is an issue called *overfitting*. Overfitting is a phenomenon such that it performs very well for training data, while the performance for test data is significantly bad. This occurs when the system is trained too precisely for the training data, it loses its generalization ability for the unknown inputs. Another case that overfitting can be expected is the quality of training data is poor due to lots of noise or unfair distribution. Then, it seems that the algorithm described in section 5.1 has high possibility to cause overfitting because the resulting separating surface is decided not to make any error over all training data, this means, the performance for the training set is always perfect. However, the simulation results show good performance for test data. It is not only because the training sets used in the simulations in 5.2 are artificially generated, noise free data, but it can be considered because the procedure which keeps the hyperplanes generated only for linearly separable subsets makes the training phase robust against noise.

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 this: is it possible to represent the non-linear separating surface by utilizing a portion of provisional curve, like parabola?

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

$$D(x) = w^t x + b \tag{6.1}$$

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

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

Staring at (6.2), this linear function can be turned into a non-linear function by performing the

following non-linear transformation on x_1 by f:

$$(x_1, x_2) \rightarrow (f(x_1), x_2) = (X_1, x_2) \tag{6.3}$$

(6.2) becomes a non-linear function for x_1

$$D(x) = w_1 f(x_1) + w_2 x_2 + b \tag{6.4}$$

However, it can also be seen as a linear function for X_1.

$$D(x) = w_1 X_1 + w_2 x_2 + b \tag{6.5}$$

Therefore, the linear separating surface $D(x)$ in (X_1, x_2) space is actually a non-linear function of f in (x_1, x_2), the original space.

According to the above observation, a non-linear separating surface can be obtained by SVM by performing the procedure Non-Linear Transformation shown in Figure 6.1.

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.

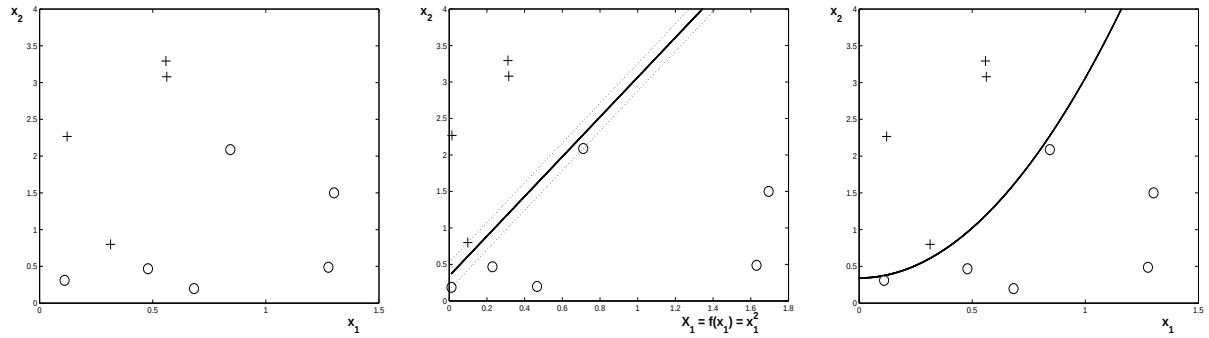
Figure 6.1: Non-Linear Transformation for obtaining a non-linear separating surface by SVM

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 separated non-linearly.



(a)Original data space (b)Transformed space (c)Original data space

Figure 6.2: Simple example of the Non-Linear Transformation

First, perform the following transformation for the attribute x_1 by $f(t) = t^2$.

$$(x_1, x_2) \rightarrow (x_1^2, x_2) = (X_1, x_2) = x \tag{6.6}$$

In the new data space (X_1, x_2), apply the linear version SVM and obtain a hyperplane $D(x)$ shown in Figure 6.2(b) and equation (6.7).

$$D(x) = w_1 X_1 + w_2 x_2 + b \tag{6.7}$$

Transforming (6.7) back into the original data space (x_1, x_2), it appears as a quadratic function in x_1 shown in Figure 6.2(c) and equation (6.8).

$$D(x) = w_1 x_1^2 + w_2 x_2 + b \tag{6.8}$$

The non-linear separating surface in the original data space is shown in Figure (6.2)(c).

The remark 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 data dimensions, 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 the non-linear transformation f, it is shown that the non-linear separating surface can be obtained by linear version of SVM without adding extra computational complexity. However, the resulting non-linear surface does not guarantee to separate the two classes correctly in the original data space. It actually happens that the linear hyperplane correctly classifies the transformed data in the transformed space, while the non-linear surface does not classify the original data correctly in the original data space.

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 separating surface by a combination of piecewise non-linear curves of selected f. The entire algorithm is shown in Figure 6.3.

First, the non-linear transformation of the selected attribute of the input data is performed by the selected f and create a new data space. 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 Iterative SVM algorithm at step *NLA2*. If the function of the optimal separating surface is selected for f, this would be the best. In general, however, the optimal separating surface is not known and this is

NLA1.	Produce the transformed input space by mapping an attribute of input data by a non-linear transformation f.
NLA2.	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.
NLA3.	Transform h_j back into the original data space. The linear hyperplanes h_j appear as non-linear surfaces g_j.
NLA4.	Infer the overall separating surface by combining g_j.

Figure 6.3: The Non-Linear Approximation Algorithm

why it is needed to be approximated.

At the step *NLA2*, linear hyperplanes are generated by applying the first three steps of the Iterative SVM algorithm, but, the hyperplanes are not combined in the transformed space since they are not guaranteed to correctly separate the data points in the original data space.

The step of forming the final separating surface is carried out after the linear hyperplanes obtained in the transformed space are back into the original data space. As illustrated in section 6.1, the linear hyperplane generated in the transformed data space appears as a non-linear surface of f.

At the final step, *NLA5*, the non-linear surfaces are combined in the original data space so as to minimize the overall training error, which is 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.

distance threshold	subset size	average # of training points	average # of hyperplanes		recognition rate (%)		
			generated	retained	max	min	average
9.5	6	89.88/100	14.98	7.43	95.00	70.50	88.99
	8	87.76/100	10.97	6.80	94.50	74.00	87.88
9.7	6	99.83/100	16.76	8.26	94.00	81.50	89.53
	8	99.71/100	12.80	7.50	93.50	84.00	89.30

Table 6.1: Simulation results for the polynomial data in Figure 5.4

6.3.1 Approximation with t^2 curve for the polynomial data

In this simulation, the polynomial separating surface in Figure 5.4 are to be approximated with the x^2 curves by transforming the x_1 coordinate by $f(t) = t^2$.

Figure 6.4(a) shows the linear hyperplanes obtained in the transformed data space (total of seventeen hyperplanes) when subset size is set to six and the distance threshold is set to 9.7. Figure 6.4(b) shows the non-linear surfaces which are transformed back to the original data space and the final separating surface combined at the step *NLA4*. The final separating surface consists of seven non-linear surfaces correctly classifies 93.5% of the test data (Figure 6.4(c)).

Other results for the simulations with the distance threshold is set to 9.5 and 9.7, each for the subset size of six and eight points 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. Also, the number of hyperplanes (actually the non-linear surfaces) retained for the final separating surface is increased in all cases. Although in this simulation taking the large threshold does not make a significant improvement of the recognition rate in average as well as in maximum, it made about 10% of improvement in the minimum performance.

distance threshold	subset size	average # of training points	average # of hyperplanes		recognition rate (%)		
			generated	retained	max	min	average
25	6	76.38/100	12.73	6.78	93.00	73.50	85.78
	8	53.36/100	6.67	4.65	93.50	69.00	84.11
26	6	93.72/100	15.62	8.17	94.00	68.50	87.38
	8	86.48/100	10.81	6.82	93.50	76.00	87.39

Table 6.2: Simulation results for the polynomial data in Figure 5.4

6.3.2 Approximation with t^3 curve for the polynomial data

Next, the experiments to utilize t^3 curve for the approximation of the polynomial separating surface for the data sets in Figure 5.4 are carried out.

In Figure 6.5, one of the results of this simulation is shown. Figure 6.5(a) shows the transformed data space and all the linear hyperplanes generated for the transformed data. Seventeen hyperplanes are generated there with the subset size is set to six and the distance threshold is 26. In Figure 6.5(b), those linear hyperplanes in the transformed space are returned to the original data space and they appear as polynomial curves of degree three. The final separating surface recognize 92% correctly over the test data (Figure 6.5(c)).

Other results are shown in Table 6.2. The distance threshold is set to 25 and 26 for the subset of size six and eight points respectively.

The overall performance became worse than the approximation with t^2 curve in the previous section, though the better performance would have been expected since the selected t^3 for f is the same degree of polynomial curve as the optimal surface. Over 50% to 70% of the generated surfaces were used to construct the final separating surface.

distance threshold	subset size	average # of training points	average # of hyperplanes		recognition rate (%)		
			generated	retained	max	min	average
8.8	6	51.60/100	8.60	5.53	89.00	37.50	77.02
	8	55.04/100	6.88	5.07	87.00	63.50	77.41
9	6	99.87/100	16.79	9.64	89.00	67.50	83.40
	8	99.79/100	12.59	8.57	89.00	72.50	82.62

Table 6.3: Simulation results for the sigmoid data in Figure 5.6

6.3.3 Approximation with t^2 curve for the sigmoid data

The Non-Linear Approximation Algorithm is applied to the sigmoid data shown in Figure 5.6. The non-linear transformation is done by $f(t) = t^2$ on the x_1 coordinate.

The linear hyperplanes generated in the transformed data space is shown in Figure 6.6(a). Seventeen hyperplanes are generated with the subsets of size six and the distance threshold 9. In Figure 6.6(b), those hyperplanes are transformed into the original data space and appear as non-linear surfaces. The final separating surface performes 84.5% on the test data (Figure 6.6(c)). As it can be seen, the resulting separating surface is almost like the one from linear approximation since any good curves were not generated (Figure 6.6(b)). The reason can be considerd because the selection of t^2 for f was not appropriate for the data in this simulation.

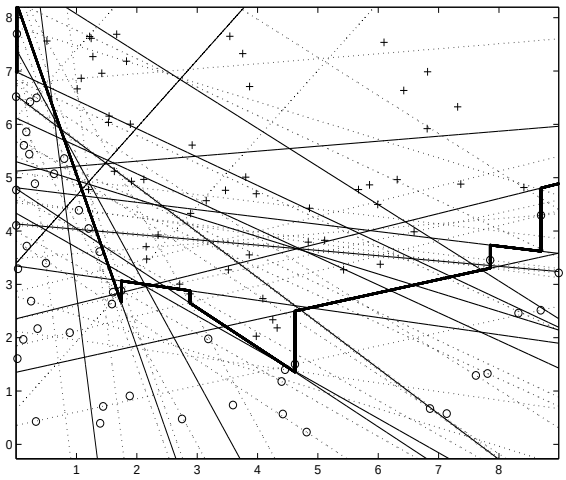
Table 6.3 shows all the simulation results with the distance threshold 8.8 and 9, the subset of size six and eight points, respectively.

The slightly better recognition rate was achieved over the linear approximation when the distance threshold 3.4 was selected. 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.

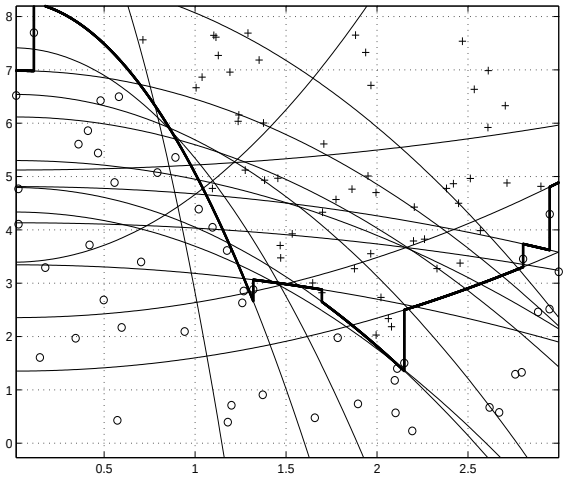
6.4 Conclusion

Comparing the simulation results in the previous section and those in section 5.2, even though a slight improvement was made for the sigmoid case, the approximation with non-linear curves does not mean 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 6.6) are compared, it is obvious that the non-linear approximation does not utilize the curves well, it is almost the same as the linear approximation. The theory to select an appropriate non-linear transformation f for the data is desired.

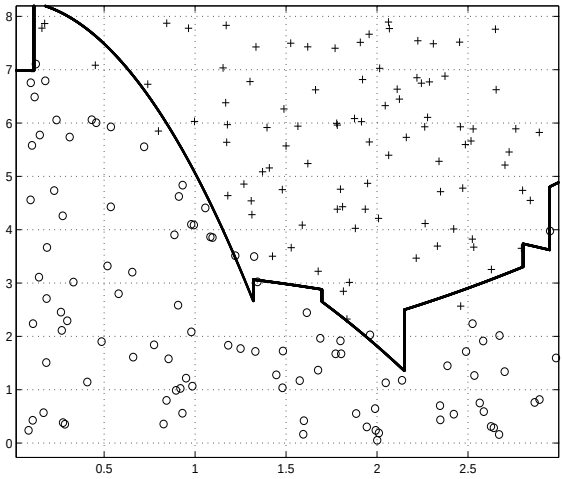
On the other hand, the impact would be big to be shown that the linear classifier SVM can produce a non-linear separating surface by introducing the non-linear transformation which does not increase the data dimensionality. Instead of employing the same procedures as the piecewise linear approximation, more appropriate algorithm to decide the final separating surface with this method would be necessary to be considered. We leave this as future work.



(a)Linear hyperplanes generated in the transformed data space

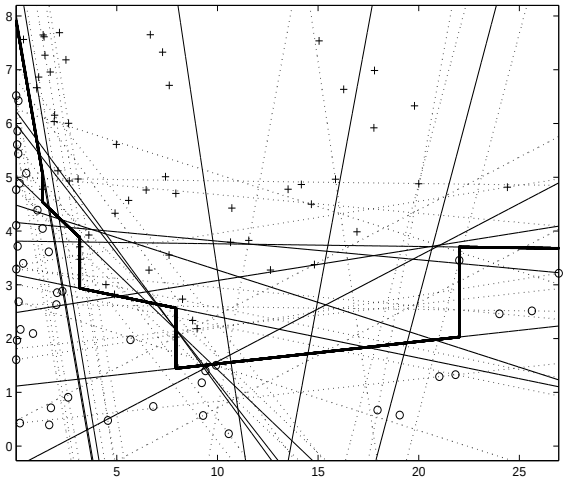


(b)Non-linear hyperplanes in the original data space

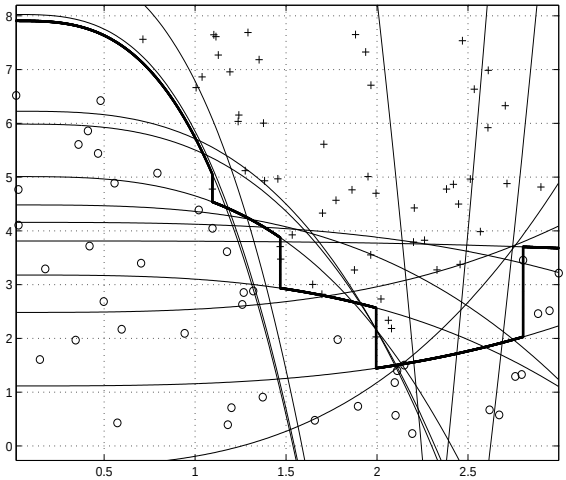


(c)Final separating surface on test data

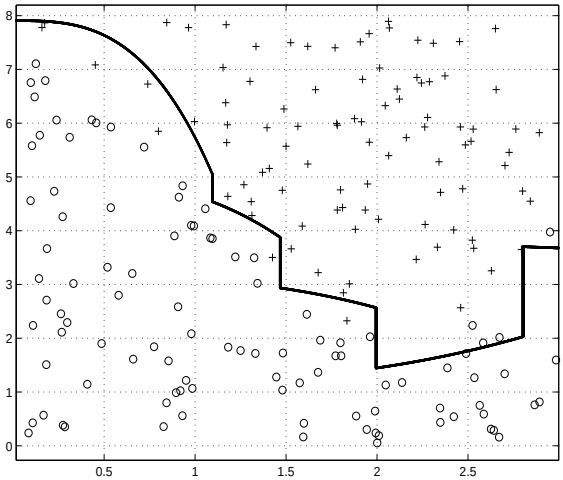
Figure 6.4: Approximation with t^2 curve for the polynomial data in Figure 5.4 with subset size = 6 and distance threshold = 9.7, 93.5% of classification accuracy for the test data



(a)Linear hyperplanes generated in the transformed data space

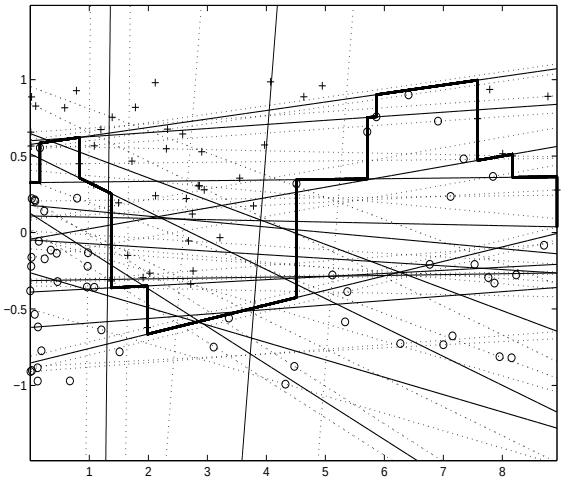


(b)Non-linear hyperplanes in the original data space

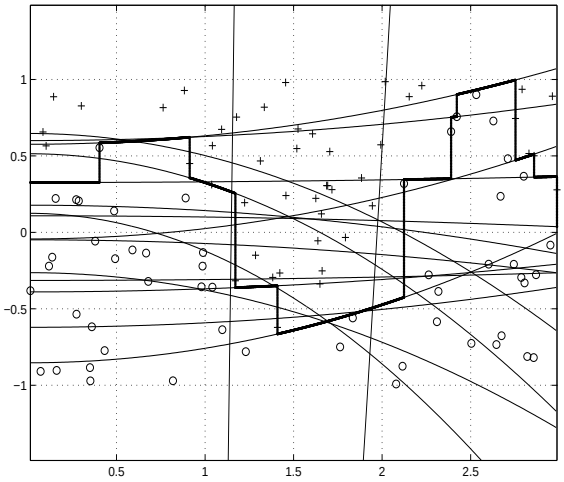


(c)Final separating surface on test data

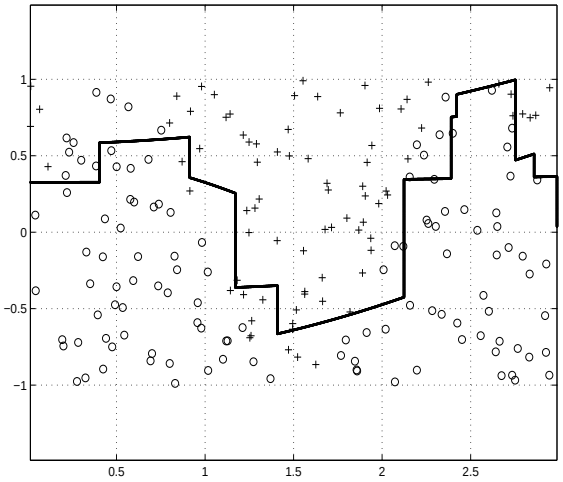
Figure 6.5: Approximation with t^3 curve for the polynomial data in Figure 5.4 with subset size = 6 and distance threshold = 26, 92% of classification accuracy for the test data



(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

Figure 6.6: An example of the simulation results for the sigmoid data in Figure 5.6 with subset size = 6 and distance threshold = 9, 84.5% of classification accuracy for the test data

Chapter 7

Conclusion and Future Works

7.1 Conclusion

This work presented a method to find the exact solution for the SVM problem by taking iterative procedure to solve the small SVM problem in small subsets of training data analytically and combine procedure to form the final separating surface that consists of piecewise linear hyperplanes. This method utilizes only the linear version SVM, especially hard margin SVM, which does not involve in working in 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 sigmoid curve. The results showed good approximations of the true separating surface.

Another contribution of this work is to have shown that SVM can generate a non-linear separating surface by performing the proposed non-linear transformation without increasing the data dimensionality. Adopting the non-linear transformation into the algorithm for the linear approximation suggests the possibility to construct an approximative surface to the true separating surface with piecewise non-linear curves. Although experimental results did not indicate the improvement over the piecewise linear approximation, more suitable algorithm for this method can be expected

to improve its performance.

7.2 Future Works

The experiments conducted in this work showed initial results with the proposed method. Various adjustments for the better performance can be considered. Also, the challenges for more difficult problems with this method can be interesting to examine. The possible directions of further research are described below.

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. Other constraints, in particular, the slopes of intermediate hyperplanes can be used to further constrain the subsets generated subsequently as to ensure that the selection is from around the true separating surface of the two classes.

7.2.2 Use of other Kernels

The hard margin SVM which is employed in this work can be seen as the hard linear kernel. The same method can be applied to other kernels, for example, to investigate the extent to which non-linearly separable data can be well separated by polynomial kernels.

7.2.3 Combine Procedure

In the method illustrated in this work, the final separating surface is formed so as not to make any error over the training points. Currently, when the transition to a new hyperplane is needed, the one which is closest to the current hyperplane is selected to avoid generating a rough surface. A smoother final surface can be obtained by considering other rules for the selection of a new

hyperplane so that the less transition would be occurred. Especially for non-linear approximation, the intersection points of those non-linear surfaces seem to be utilized for generating a smooth separating surface. Also, allowing some errors over the training points in the combine procedure may improve its performance for the test data.

7.2.4 Approximations of difficult class boundary

The proposed method has exhibited the validity for the cases with non-linear class boundary, which is one of the difficult problems in classification. The next challenge would be for the more difficult cases, like approximation of closed polyhedral class boundary or distinct class boundary as shown in Figure 2.1(d)(e). For those problems, the current method needs to be modified to be able to select and keep more than one hyperplanes within a region.

Bibliography

[1] Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach.* Prentice Hall, 1995.

[2] Tom M. Mitchell. *Machine Learning.* The McGraw-Hill Science Companies, 1997.

[3] V. Vapnik. Estimation of dependencies based on empirical data. *Springer Verlag*, 1982.

[4] V. Vapnik. The nature of statistical learning theory. *Springer Verlag*, 1995.

[5] Shigeo Abe. *Support Vector Machines for Pattern Classification.* Kobe University, 2002.

[6] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other Kernel-based Learning Methods.* Cambridge University Press, 2000.

[7] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[8] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. *Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing*, pages 276–285, 1997.

[9] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research Technical Report MSR-TR-98-14, 1998.

[10] Thorsten Joachims. Making large-scale svm learning practical. Technical report, LS8-Report, 24, University of Dortmund, 1998.

[11] Rinako Kamei and Anca Ralescu. Piecewise linear separability using support vector machines. *Proceedings of the 14th Midwest Artificial Intelligence and Cognitive Science Conference*, pages 52–56, 2003.

[12] Shun-ichi Amari and Si Wu. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12:783–789, 1999.

[13] Shun-ichi Amari and Si Wu. Conformal transformation of kernel functions: A data-dependent way to improve support vector classifiers. *Neural Processing Letters*, 15(1):59–67, 2001.

[14] Christopher J. C. Burges. Simplified support vector decision rules. *Proceedings of the 13th International Conference on Machine Learning*, pages 71–77, 1996.