University of Alberta

A Support Vector Machine Model for Pipe Crack Size Classification

by

Chuxiong Miao

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Fall 2009
Edmonton, Alberta

Examining Committee

Ming J. Zuo, Mechanical Engineering

Xiaodong Wang, Mechanical Engineering

Marinal Mandal, Electrical and Computer Engineering

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Oil and gas are transported mainly through pipelines. As time goes by, damages and defects may develop in these pipelines. If the defects cannot be detected in time, the property loss and environmental damage could be tremendous. For example, on August 7, 2006, BP p.l.c.[1] began shutting down the U.S.A.'s largest oil field due to heavy corrosion and a small leak detected in a critical pipeline in its Prudhoe Bay operation in Alaska. Reporters said this made the prices at the gas pump begin to rise by as much as 5 cents a gallon in some cities because the Prudhoe Bay field produces 400,000 barrels a day, which was 8 percent of American crude at that time [1]. Clearly, periodic inspection of pipelines is necessary to guarantee they are in good working order [2].

Ultrasonic testing is very popular in the pipeline industry. NDT (Nondestructive

[1]Previously known as British Petroleum and the third largest global energy company in 2006. BP is a multinational oil company with headquarters in London. The official website is http://www.bp.com

Testing) test uses high frequency, highly directional sound waves to measure material thickness, find hidden internal flaws, or analyze material properties in metals, plastics, composites, ceramics, rubber, and glass. Using frequencies beyond the limit of human hearing, ultrasonic instruments generate short bursts of sound energy that are coupled into the test piece; the instrument then monitors and analyzes reflected or transmitted wave patterns to generate test results [3].

Researchers have used ultrasonic data to classify defect types and defect sizes for pipeline fault analysis. Fei *et al.* [4] tried to classify different kinds of flaws, such as circular, rectangular, cylindrical and welded defects, using ultrasonic signals from the seafloor petroleum-transporting pipeline. Sinha *et al.* [5] classified the pipeline's surface defects such as holes and cracks using the neuro-fuzzy technique. Ravanbod [6] categorized pipeline flaws as being of four types: internal, external, inside the pipe, and both internal and external. As well, he classified these flaws according to size, as shallow, medium and deep. Zhao *et al.* [7] classified mechanical dents in pipelines into two categories, cup dents and saucer dents and also estimated their size. Murigendrappa *et al.* [8] predicted crack size in pipelines filled with fluid. The error in crack-size prediction lies in the range of -16.44% to 10.30% for aluminium, and -5.83% to 12.04% for mild steel.

Information on crack sizes in pipelines could help in making decisions on what kind of maintenance strategy will be used. For example, if the size of a crack in a pipeline exceeds some value, replacement may be done right away. If, however, it does not, this could be considered safe. From this point of view, classifying crack sizes correctly is an interesting research topic. This thesis focuses on classification of crack sizes in pipelines.

To achieve better performance as per lower test error rates in classification, artificial intelligence technology such as Neural Network (NN) and Support Vector

Machine (SVM) have been employed. Neural Networks have been used to solve a variety of classification problems involving speech recognition [9], finger print recognition [10], handwritten character recognition [11], ultrasonic signal classification [5, 6] and more, but Neural Networks may converge to locally optimal solutions [12]. In contrast, convexity is an important and interesting property of the Support Vector Machine which was arouse out of the Statistical Learning Theory developed by Vladimir Vapnik and co-workers at AT&T Bell Laboratories in 1995. The reason for its importance is that SVM theory is based on the principle of structural risk minimization (SRM) while Neural Network is based on empirical risk minimization. Because of the SRM principle, the global optimal value of SVM can be obtained; on the other hand, Neural Network may obtain only a local optimal value [15]. In addition, unlike the Neural Network method, the SVM approach does not attempt to control model complexity by keeping the number of features small. Support Vector Machines marked the beginning of a new era in the learning-from-examples paradigm, which has been developed recently [16]. That's why we believe that the SVM is the appropriate tool for classifying the pipelines' cracks according to size.

The problem in classifying the size of a pipeline's crack is that we must categorize the ultrasonic data correctly as much as possible. To simplify the problem, we consider only the binary class situation; this means we will have only two categories, i.e., large and small. Multiple class problems may be solved by successive application of the binary class approach. Based on the ultrasonic data collected, a classifier will be built. Whenever new ultrasonic data comes in, it can be used to predict whether the crack is large or small. In this thesis, we're trying to find an appropriate classification model that uses the SVM as the classifier in the analysis of pipe ultrasonic data regarding cracks.

1.2 Research Objective

In this thesis, we will use a binary class-data-dependent SVM with a Gaussian kernel for the ultrasonic pipe crack data, in order to correctly categorize the new data as much as possible. Solving such a problem involves three steps: collecting raw ultrasonic data, preprocessing these collected data, and building and tuning the classifier. This thesis focus on the third step, building and tuning the classifier. The research objectives are as follows:

1. Find an alternative indicator for selecting the parameters of the SVM classifier.

2. Construct a data dependent kernel, based on the basic Gaussian kernel.

1.3 Organization of Thesis

This thesis is organized as follows. Chapter 2 will introduce the fundamental algorithm of the Support Vector Machine, because we will use a great deal of terminology from this chapter in later discussion. This chapter introduces both hard margin and soft margin SVM, and discusses important concepts like structural risk, VC dimension and "kernel trick". It also introduces the parameters of the SVM classifiers. Chapter 3 mainly discusses the data preprocessing, which includes feature extraction of the ultrasonic pipe crack data using digital signal processing technology, and feature selection using both the sequential backward selection method (SBS) and the sequential forward selection method (SFS). After the data preprocessing, we create the so called "Optimal B-Scan feature data" as the input for later discussion. Existing algorithms are used in this chapter. Chapter 4 use a proposed indicator called "KFD Ratio" for selecting suitable SVM parameters. Because our input from "Optimal B-Scan feature data sets" includes 9 specimens of crack size (from 0mm

to 3mm), different separations of the data are used to test the generalization performance of the classifier. Chapter 5 adopts a data dependent method to construct a more complicated kernel based on the Gaussian kernel the parameters of which are determined in Chapter 4. This kernel is used to improve our generalization ability, and some experiments are done to prove its efficiency. Chapter 6, the final chapter, presents our summary and conclusions. As well, possible areas for future research work are suggested.

Chapter 2

Fundamentals of SVM Classification

Support Vector Machine (SVM) is a supervised learning method used for classification and regression. Rooted in the statistical learning theory developed by Vladimir Vapnik and co-workers at AT&T Bell Laboratories, SVM quickly received attention from the pattern recognition community due to its theoretical and computational merits which include, its simple geometrical interpretation of the margin, the uniqueness of the solution, the statistical robustness of the loss function, the modularity of the kernel function, and the overfit control obtained through its choice of a single regularization parameter [16]. These technical terms are useful for the understanding of this thesis and will be further discussed in this chapter.

This chapter will summarize SVM fundamental concepts and methods for reference in later chapters. This chapter is organized as follows. Section 2.1 introduces the binary classification problem. Section 2.2 introduces the concept of structural risk minimization which enables SVM to be a high performance classifier. Finally, Section 2.3 presents the algorithm of SVM for binary classification. This section also

includes concepts such as the optimal hyperplane and support vectors. In addition, it discuss the regularization parameter, C, of SVM and kernel functions.

2.1 Binary Classification Problems

Suppose we are given m observations. Each observation consists of a pair: an input vector, $\boldsymbol{x}_i \in \mathbb{R}^n, i = 1, \cdots, m$, and the associated label, y_i. These given data can be summarized as

$$(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_m, y_m) \in \mathcal{X} \times \{\pm 1\} \tag{2.1}$$

where $\mathcal{X}$ is some nonempty set of n dimensions. There are only two classes, and these are labeled by $+1$ and -1, respectively. We run the learning algorithm of SVM to train a classifier using these observations. The observations used for training the classifier are called **training data**. The task of the learning algorithm of SVM is to learn the mapping, $\boldsymbol{x}_i \mapsto y_i$, and find a function, f, that will correctly classify a new observation, $(\boldsymbol{x}, y)$, so that $f(\boldsymbol{x}) = y$ [17]. Usually, the performance of the algorithm will be measured on **test data** that is independent of the training data. A 2-dimensional example ($\boldsymbol{x}_i \in \mathbb{R}^2$) of binary classification is shown in Figure 2.1, where diamonds and circles represent the two classes. With SVM, we get the decision boundary of the classes. When a new input vector is sent in, we can easily allocate it to one of the two classes.

Compared to other learning algorithms such as Neural Network, SVM achieves a better balance between training error and complexity when solving binary classification problems. This is because it applies the principle of structure risk minimization [15]. We will use a binary classification problem as an example to explain the concepts of training error and model complexity, and to discuss structural risk minimization.

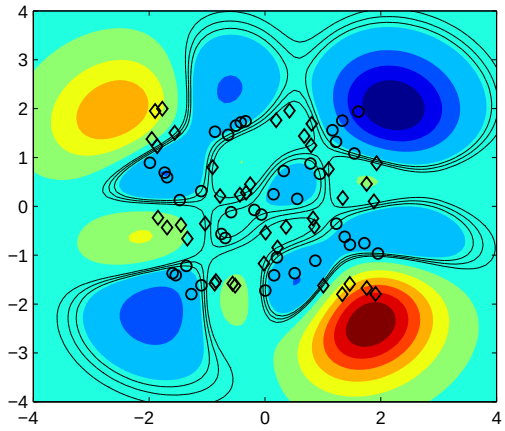


Figure 2.1: A 2D example of binary classification.

2.2 Structural Risk Minimization (SRM)

Using training data, we can minimize the average **training error**, which is also called the **empirical risk**. The definition of the empirical risk for binary classification is,

$$R_{emp}(f) = \frac{1}{2m} \sum_{i=1}^{m} |f(\boldsymbol{x}_i) - y_i| \tag{2.2}$$

The VC (Vapnik-Chervonenkis) theory shows that it is necessary to restrict the set of functions from which f is chosen so that it has a capacity (to be defined next) suitable to the amount of available training data [17]. This theory provides bounds on the **test error**, which is the expectation error, $R(f)$, of test data. The minimization of these bounds, which depend on both the empirical risk and the capacity of the function class, f, leads to the principle of **structural risk minimization (SRM)** [18].

The VC dimension is a measure of the capacity mentioned above, which can be explained as follows: each class of a mapping separates the input vectors in a certain way and thus induces a certain labeling of them. Since the labels are between $\{\pm 1\}$, there are at most 2^m separations [17]. The VC dimension is defined as the largest m

for which there exists a set of m points that the class of function can separate. For example, Figure 2.2 shows that for three two-dimensional points, all 8 possibilities of separation can be realized using a straight line. This would not work if we were given 4 points. Thus the VC dimension of the class of separating straight lines in $\mathbb{R}^2$ is 3. The straight line in Figure 2.2 is called a "**hyperplane**" when the concept is applied to 3-dimensional space and beyond. For example, in 3-dimensional space, the hyperplane is an ordinary plane which divides the space into two half-spaces.

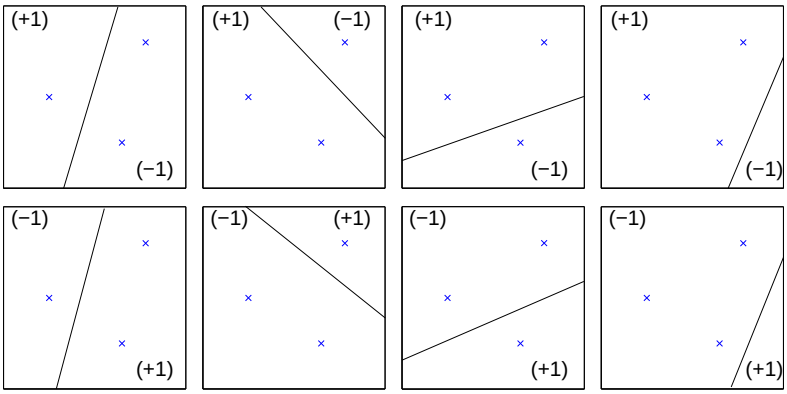


Figure 2.2: VC dimension = 3 in $\mathbb{R}^2$.

From statistical learning theory , there is a bound which always holds for the structural risk, i.e., the test error [15]. The principle of SRM is to minimize the bounds.

$$R(f) \leq R_{emp}(f) + \phi(h, m, \delta) \tag{2.3}$$

where the confidence term (or the capacity term) ϕ is defined as

$$\phi(h, m, \delta) = \sqrt{\frac{1}{m}(h(\ln \frac{2m}{h} + 1) + \ln \frac{4}{\delta})}. \tag{2.4}$$

Here, δ is the significance level, h is the VC dimension, and m is the size of the training data. The confidence term in Eq (2.4) increases monotonically with h. Given

a set of finite training data, we can always find a learning machine which achieves a training error of zero, (providing we have no input vectors contradicting each other, i.e., whenever two input vectors are identical, they must come with the same label). To correctly separate arbitrary training data, this machine will necessarily require a large VC dimension, h. The bound in Eq (2.3) shows that a small training error does not guarantee a small test error.

The support vector machine achieves the goal of the SRM principle, minimizing the bound on the VC-dimension and the training error concurrently, by a completely automatic optimization procedure [12]. Below, we will discuss the algorithm of SVM and show how SVM implements the SRM principle.

2.3 Algorithms for SVM Classification

Suppose we have m $n-$dimensional training data, $\mathcal{X}$, which meet the requirements in Eq (2.1). A linear decision function can be defined as [13]:

$$\boldsymbol{w}^T\boldsymbol{x} + b = 0 \tag{2.5}$$

where $\boldsymbol{w}$ is an n-dimensional vector and b is a bias term. When two sets of points in a 2-dimensional space can be completely separated by a single straight line, they are said to be **linearly separable**. In general, two classes are linearly separable in n-dimensional space if they can be separated by an $n-1$ dimensional hyperplane. If the training data are linearly separable, a linear decision function, $\boldsymbol{w}^T\boldsymbol{x} + b = 0$, exists which meets

$$\boldsymbol{w}^T\boldsymbol{x}_i + b \begin{cases} > 0 \quad for \quad y_i = 1, \\ < 0 \quad for \quad y_i = -1. \end{cases} \qquad i = 1, \cdots, m \tag{2.6}$$

Because the training data are linearly separable, no training data satisfy $\boldsymbol{w}^T\boldsymbol{x} + b = 0$. So, to control separability, instead of Eq (2.6), we consider the following

inequalities:

$$\boldsymbol{w}^T\boldsymbol{x}_i + b \begin{cases} \geqslant +1 & for \quad y_i = 1, \\ \leqslant -1 & for \quad y_i = -1. \end{cases} \qquad i = 1, \cdots, m \tag{2.7}$$

Here, $+1$ and -1 on the right-hand sides of the inequalities can be a constant c (> 0) and $-c$, respectively. But by dividing both sides of the inequalities by c, Eq (2.7) is obtained. Eq (2.7) can be expressed compactly with the following inequalities [13]:

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 \geqslant 0, \qquad i = 1, \cdots, m \tag{2.8}$$

Figure 2.3 gives a 2-dimensional linearly separable example. The distance between the hyperplane $\boldsymbol{w}^T\boldsymbol{x} + b = 0$, a solid straight line, and the origin is $\frac{-b}{\|\boldsymbol{w}\|}$. The distance between the hyperplanes $\boldsymbol{w}^T\boldsymbol{x} + b = 1$ and $\boldsymbol{w}^T\boldsymbol{x} + b = -1$, which is the dashed straight lines, to the hyperplane $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ are both $\frac{1}{\|\boldsymbol{w}\|}$. Thus the distance between two dashed hyperplanes is $\frac{2}{\|\boldsymbol{w}\|}$. We call this distance the **margin**. Also, the data that satisfy the equalities in Eq (2.8) are called **support vectors**. In Figure 2.3, the data corresponding to the filled circles and the filled rectangle are support vectors. Note that all the hyperplanes are parallel.

Figure 2.4 shows two decision functions, the blue solid line and the pink solid line. They both satisfy Eq (2.8). In fact, there are an infinite number of decision functions that satisfy Eq (2.8). In geometry, the decision functions are called separating hyperplanes, and the hyperplane with the maximum margin is called the **optimal separating hyperplane** [13]. The classifier whose decision function corresponds to the optimal separating hyperplane has the best **generalization ability** [13]. A classifier is said to have good generalization ability if it performs on test data almost as well as it does on the training data [14].

To get the optimal separating hyperplane, we need only find the maximum margin by minimizing $\|\boldsymbol{w}\|^2$, subject to Eq (2.8). Thus, the optimal separating hyperplane

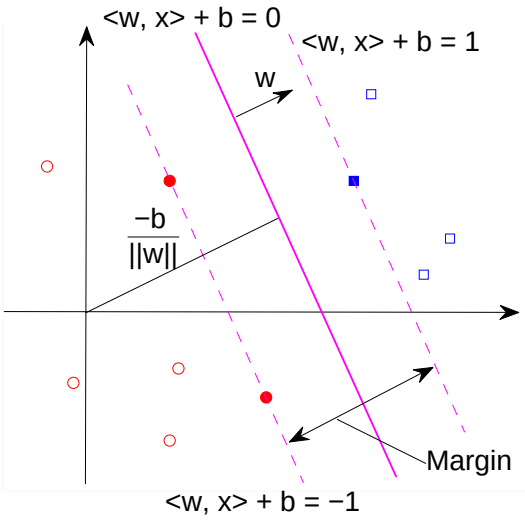


Figure 2.3: Linearly separable hyperplane

problem can be summarized as follows

$$\begin{aligned} &\min \ \ \tfrac{1}{2}\|\boldsymbol{w}\|^2 \\ &Subject\ to: \\ &\qquad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 \geqslant 0 \qquad \forall i. \end{aligned} \tag{2.9}$$

Unlike the case with linearly separable data, we can't find an $n-1$ dimensional hyperplane that can separate two classes in an n-dimensional space. This kind of data is called non-linearly separable. Figure 2.5 shows the non-linearly separable data in 2-dimensional space where any single straight line can't separate the two classes marked as circles and squares. By introducing a group of non-negative slack variables, ξ_i ($\geqslant 0$), we can apply the optimal separating hyperplane problem to the case with non-linear separable data [21]. Eq (2.8) then becomes [13]:

$$\begin{cases} y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geqslant 1 - \xi_i \\ \xi_i \geqslant 0 \end{cases} \qquad \forall i. \tag{2.10}$$

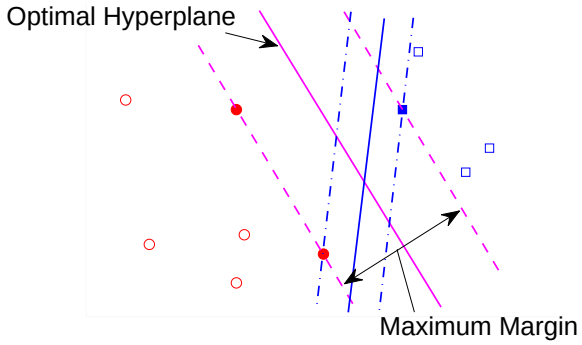


Figure 2.4: Optimal separating hyperplanes

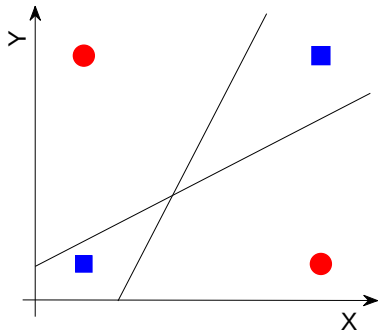


Figure 2.5: Non-linear separable data in $\mathbb{R}^2$

For the training data, $\boldsymbol{x}_i$, if $0 < \xi_i < 1$ (the circle with a line connected to the hyperplane in Figure 2.6), the data do not have the maximum margin but are still correctly classified. In other words, this circle is classified as the circle class. But if $\xi_i \geqslant 1$ (the square with a line connected to the hyperplane in Figure 2.6) the data is misclassified by the optimal separating hyperplane. This square should be put in the circle class. In shorts, for an error to occur, the corresponding ξ_i must exceed unity. Hence a natural way to assign an extra cost for errors is to change the objective function, minimizing it from $\frac{\|\boldsymbol{w}\|^2}{2}$ to $\frac{\|\boldsymbol{w}\|^2}{2} + C\sum_{i=1}^{m}\xi_i$, where C is a parameter to be chosen by the user, with a larger C corresponding to a higher penalty being assigned

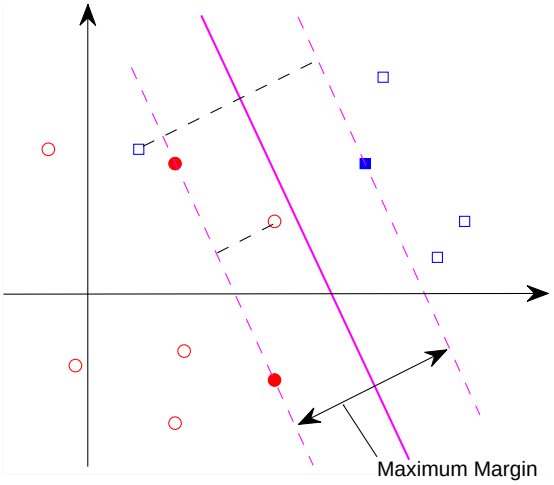


Figure 2.6: Non-linear separable case in a 2-dimensional space

to errors [22]. By regulating parameter C, which sometimes is called the trade-off parameter, we implement the principle of structural risk minimization, that is, we minimize the bound on the VC-dimension and the training error concurrently. The optimal separating hyperplane for a non-linearly separable problem can be summarized as

$$\begin{aligned} &\min \ \ \tfrac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ &\textit{Subject to}: \\ &\quad \begin{cases} y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 + \xi_i \geqslant 0 \\ \xi_i \geqslant 0 \end{cases} \qquad \forall i. \end{aligned} \tag{2.11}$$

We can reformulate Eq (2.11) using the Lagrangian formulation [22]. Following the rule of forming Lagrangian, the $\geqslant 0$ constraints should be multiplied by positive Lagrangian multipliers and subtracted from the objective function. By introducing the non-negative Lagrangian multipliers α_i and β_i $(i = 1, \cdots, m)$, we get the

Lagrangian equation:

$$\begin{aligned} L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \quad & \tfrac{\|\boldsymbol{w}\|^2}{2} + C \sum_{i=1}^{m} \xi_i \\ & - \sum_{i=1}^{m} \alpha_i [y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{m} \beta_i \xi_i. \end{aligned} \tag{2.12}$$

The Lagrangian L has to be minimized with respect to the primal variables, $\boldsymbol{w}$ and b, and maximized with respect to the dual variables, α_i and β_i [17]. In other words, a saddle point has to be found. By applying the Karush-Kuhn-Tucker (KKT) condition [13], we get the following group of equations:

$$\begin{aligned} \tfrac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i = 0 \\ \tfrac{\partial L}{\partial b} = -\sum_{i=1}^{m} \alpha_i y_i = 0 \\ \tfrac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \\ y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1 + \xi_i \geqslant 0 \\ \xi_i \geqslant 0 \qquad\qquad where \;\; i = 1, \cdots, m. \\ \alpha_i \geqslant 0 \\ \beta_i \geqslant 0 \\ \alpha_i [y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1 + \xi_i] = 0 \\ \beta_i \xi_i = 0 \end{aligned} \tag{2.13}$$

We can draw some conclusions from Eq (2.13) [13].

- If $\alpha_i < C$, then $\xi_i = 0$. We can simply take any point for which $0 < \alpha_i < C$ and use $\alpha_i [y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1 + \xi_i] = 0$ $(with\ \xi_i = 0)$ to compute b. If $\alpha_i = 0$, then $\boldsymbol{x}_i$ is correctly classified. If $0 < \alpha_i < C$, then $y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1 + \xi_i = 0$ and $\xi_i = 0$ [13].

- If $\alpha_i = C$, then $y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1 + \xi_i = 0$ and $\xi_i \geqslant 0$ [13].

- If $0 \leqslant \xi_i < 1$, then $\boldsymbol{x}_i$ is correctly classified, and if $\xi_i \geqslant 1$, then $\boldsymbol{x}_i$ is misclassified.

The KKT condition helps to reconstruct the primal problem in Eq (2.12) making it a **dual optimization problem** which eliminates the primal variables $\boldsymbol{w}$ and b [22]. The dual problem of determining the hyperplane decision function can be written as,

$$\begin{aligned} &\max L(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \tfrac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \\ &\textit{Subject to} \\ &\qquad 0 \leqslant \alpha_i \leqslant C \quad \forall i \\ &\qquad \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned} \tag{2.14}$$

where $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ represents the dot products of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. As we can see, in Eq (2.14) all the training data appear in the form of dot products.

In a Support Vector Machine the optimal hyperplane is determined so as to minimize the test error rate. Whether the training data are linearly separable or not, the separating hyperplane is determined optimally by Eq (2.14). The obtained classifier may not have a low error rate even if the hyperplane is determined optimally [13]. To lower the error rate, one can map the input vector of the training data to a new higher dimensional space, called the **feature space**, by doing a non-linear transformation using suitably chosen basis functions. This is known as the "**kernel trick**". In a mathematical way, we use a function, ϕ, which maps the input vectors to feature space $\mathcal{H}$, causing the training program to depend on the data, $\phi(\boldsymbol{x})$, in feature space $\mathcal{H}$. The mathematic form for the mapping is

$$\phi : \mathcal{X} \mapsto \mathcal{H} \tag{2.15}$$

where $\mathcal{X} \in \mathbb{R}^n$ and the dimension of feature space $\mathcal{H}$ depend on the non-linear transformation function, ϕ. An example of the dimension of $\mathcal{H}$ space is given late in this section. Usually, the dimension of feature space $\mathcal{H}$ is much higher than the dimension of the input vectors, $\mathcal{X}$. Since we use $\phi(\boldsymbol{x})$ instead of $\boldsymbol{x}$ as the input, the

dual problem of determining the hyperplane decision function can be written as

$$max\ L(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle. \tag{2.16}$$

If there is a "**kernel function**", k, such that $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$, we need to use the function, k, only in the training algorithm, never needing to know explicitly what ϕ is [22].

For example, Table 2.1 shows the map function, $\phi(\boldsymbol{x})$, so as to explicitly verify the following homogeneous polynomial[1] kernel,

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle)^2 = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle.$$

In paper [22], Burges proves that the corresponding feature space, $\mathcal{H}$, is a Euclidean space of dimension $\binom{n+d-1}{d}$, where n is the dimension of the input vector, $\mathcal{X}$, (in Table 2.1, $n = 2\ and\ 3$) and d is the degree of the homogeneous polynomial (here $d = 2$). If $n = 256,\ d = 2$, making the dimension of $\mathcal{H}$ 183,181,376. Thus the feature space, $\mathcal{H}$, is high dimensional space with an enormous number of dimensions. Sometimes there are an infinite number of these dimensions. One example is the gaussian kernel, $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2})$ [23][24][20].

For the above example, the polynomial kernel enables us to work in the space spanned by dot products of any $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$ values, provided that we are able to do our work solely in terms of dot products, without any explicit use of a mapped function, $\phi(\boldsymbol{x})$. Using the kernel function, the input vectors, $\mathcal{X}$, are mapped into higher dimensional feature space, $\mathcal{H}$, and the mapping function, $\phi(\boldsymbol{x})$, need not be calculated explicitly. Thus we can take higher-order statistics into account without a combinatorial explosion of time and memory complexity [17].

[1] In mathematics, a homogeneous polynomial is a polynomial whose terms are monomials all having the same total degree; or are elements of the same dimension. For example, $x^5+2x^3y^2+9x^1y^4$ is a homogeneous polynomial of degree 5.

Table 2.1: $\mathcal{X} \mapsto \mathcal{H}$ for Homogeneous Polynomial Kernel with a Degree $d = 2$

Input Space	$\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbf{R}^2$	$\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbf{R}^3$
$\boldsymbol{x}_i$	$(\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2})$	$(\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}, \boldsymbol{x}_{i,3})$
$\boldsymbol{x}_j$	$(\boldsymbol{x}_{j,1}, \boldsymbol{x}_{j,2})$	$(\boldsymbol{x}_{j,1}, \boldsymbol{x}_{j,2}, \boldsymbol{x}_{j,3})$
Mapped Space	$\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \in \mathbf{R}^3$	$\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \in \mathbf{R}^6$
$\phi(\boldsymbol{x}_i)$	$(\boldsymbol{x}_{i,1}^2, \sqrt{2}\boldsymbol{x}_{i,1}\boldsymbol{x}_{i,2}, \boldsymbol{x}_{i,2}^2)$	$(\boldsymbol{x}_{i,1}^2, \boldsymbol{x}_{i,2}^2, \boldsymbol{x}_{i,3}^2, \sqrt{2}\boldsymbol{x}_{i,1}\boldsymbol{x}_{i,2}, \sqrt{2}\boldsymbol{x}_{i,1}\boldsymbol{x}_{i,3}, \sqrt{2}\boldsymbol{x}_{i,2}\boldsymbol{x}_{i,3})$
$\phi(\boldsymbol{x}_j)$	$(\boldsymbol{x}_{j,1}^2, \sqrt{2}\boldsymbol{x}_{j,1}\boldsymbol{x}_{j,2}, \boldsymbol{x}_{j,2}^2)$	$(\boldsymbol{x}_{j,1}^2, \boldsymbol{x}_{j,2}^2, \boldsymbol{x}_{j,3}^2, \sqrt{2}\boldsymbol{x}_{j,1}\boldsymbol{x}_{j,2}, \sqrt{2}\boldsymbol{x}_{j,1}\boldsymbol{x}_{j,3}, \sqrt{2}\boldsymbol{x}_{j,2}\boldsymbol{x}_{j,3})$
Kernel	$(\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle)^2$	$(\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle)^2$
$k(\boldsymbol{x}_i, \boldsymbol{x}_j)$	$= \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$	$= \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$

Not all kernels can be expressed in the form of $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$, but the kernels which meet **Mercer's condition** always hold this property [22]. In the following, if there is no confusion, we simply call it the kernel. Table 2.2 shows some popular kernels and their properties.

Table 2.2: List of kernels and their properties

Kernel	Formula	Properties
Polynomial	$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^d$	Homogeneous
	$k(\boldsymbol{x}, \boldsymbol{x}') = (\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + c)^d$	Not homogeneous
Gaussian	$k(\boldsymbol{x}, \boldsymbol{x}') = exp(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\sigma^2})$	Radial basis function[1] (RBF)
Sigmoid	$k(\boldsymbol{x}, \boldsymbol{x}') = \tanh(\kappa \langle \boldsymbol{x}, \boldsymbol{x}' \rangle + \vartheta)$	$\kappa > 0$ *and* $\vartheta < 0$

1. A radial basis function (RBF) is a real function whose value depends only on its distance from the origin, so that $\phi(\boldsymbol{x}) = \phi(\|\boldsymbol{x}\|)$; or, alternatively, on the distance from some other point, c, called a center, so that $\phi(\boldsymbol{x}, c) = \phi(\|\boldsymbol{x} - c\|)$.

Although there are many kernels that can be used for the SVM algorithm, in this

thesis, we discuss only the application of the Gaussian kernel. More details will be provided in Chapter 4.

2.4 Extensions

In this chapter, we have briefly discussed the algorithm of SVM based on binary classification problems.

For those who are interested in the algorithm of SVM and want to obtain more information on matters such as the KKT condition, Mercer's condition, SVM regression and SVM multi-classification, please consult the references [13, 15, 17, 22].

Chapter 3

Ultrasonic Data Preprocessing for Pipe Crack Classification

In SVM classification, data preprocessing transforms the raw data into a format that can be more easily and effectively used. There are different tools and methods used for preprocessing. Kaastra *et al.* [25] smooth both input and output data by using either simple or exponential moving averages for forecasting financial and economic time series. Kaper *et al.* [26] apply the bandpass filter to the raw data and normalize them to an interval of $[-1, +1]$ for classifying electroencephalogram (EEG)-signals to detect the absence or presence of the P300 component in EEG event related potentials, crucial for the P300 speller paradigm in Brain-Computer Interfacing. Crone *et al.* [27] do data reduction for evaluation of classifier sensitivity in marketing by means of feature selection which aims at identifying the most relevant, explanatory input variables within a data set.

One objective of this thesis is to find a good alternative indicator for the SVM classifier's parameter selection. To reach this goal, preprocessed input vectors are needed. This Chapter prepare these input vectors for later use. Section 3.1 introduces

the raw data collection. Based on the collected raw data, two preliminary data preprocessing techniques, average and gating [28], are applied. Section 3.2 applies signal processing techniques to extracting features, forming two feature data sets. Finally, Section 3.3 uses two reported methods, sequential forward selection and sequential backward selection [13], to preprocess the formed feature data sets and reduce their dimensions.

3.1 Raw Data Collection

3.1.1 Introduction of the Ultrasonic Test

In an ultrasonic test, high frequency sound waves (ultrasound) are introduced into a test object, and the corresponding responses are collected for further analysis. Useful information about the test object (such as material properties, geometric dimensions, and flaw size and location) can be identified by analyzing the received ultrasonic signals [29][30].

Generally, ultrasonic tests can be classified as being either the transmission technique or the reflection technique [31]. Figure 3.1 [31] shows a transmission ultrasonic test. A transmitter sends ultrasound through a medium and a receiver detects the amount that has reached it on the other side. Imperfections or other conditions in the medium between the transmitter and receiver reveal their presence by reducing the amount of ultrasound which passes through [32]. In a reflection (or pulse-echo) ultrasonic test, the transducer performs both the sending and the receiving of the pulsed wave as the ultrasound is reflected back to the device. Figure 3.2 shows the principle of the pulse-echo ultrasonic test. The transducer sends a pulse to the test object and an echo (reflected ultrasound) returns from an interface, such as the back wall of the test object, or from an imperfection within the object, such as a crack.

The echo can be displayed as a signal with an amplitude representing the intensity of the reflection, and the distance representing the arrival time of the reflection. At the same time, the echo can be collected in a number of different formats. The two most common formats are known as A-Scan and B-Scan presentations.

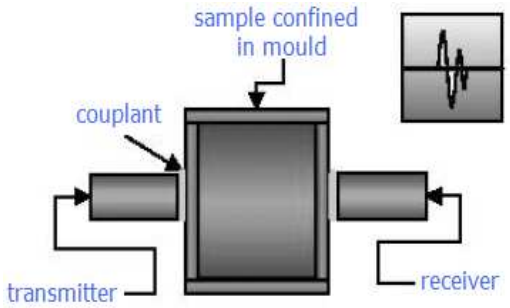


Figure 3.1: Principle of the transmission ultrasonic test [31]

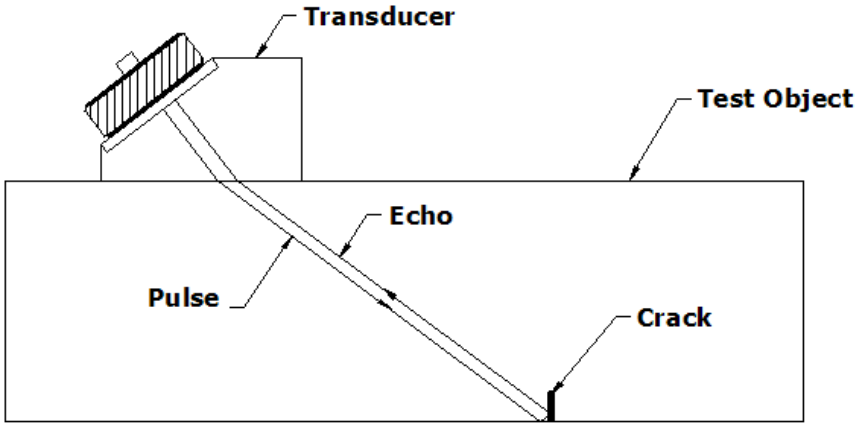


Figure 3.2: Principle of the pulse-echo ultrasonic test

A-Scan (see Figure 3.6(b)), in which echo amplitude and time are plotted on a simple grid with the horizontal axis representing time and the vertical axis representing amplitude, is the most basic presentation of ultrasonic waveform data [34]. The **B-Scan** presentation (see Figure 3.6(a)) gives a profile (cross-sectional) view of the

test specimen [35]. In it, time is displayed along the horizontal axis and the linear position of the transducer is displayed along the vertical axis. The echo amplitude is usually displayed using a third axis (the ultrasonic data is plotted in 3−dimensional space) or in different colors which represent the intensity of the signal (the ultrasonic data is plotted in 2−dimensional space).

This thesis uses the pulse-echo ultrasonic test to obtain the ultrasonic data (the reflected ultrasound or echo), and those data are saved using both A-Scan and B-Scan presentations. The setup of the experiment system for collecting the ultrasonic data is introduced below.

3.1.2 The Setup of the Experimental System

Figure 3.3 [36] shows the schematic of the complete experimental system, which consists of the following major subsystems: the ultrasonic sensor, the specimen, the Bi-slide linear position system, the OmniScan UT unit, and the fixtures [36].

The ultrasonic transducer used is the Kraukramer Benchmark Series, miniature angle beam transducer (2.25MHz, 0.5" element diameter and 45° wedge).

The specimens are nine 4140-steel blocks with electrical discharge machining (EDM) slots of different depths prepared by a machine shop. These nine specimens have the following slot depths (given in mm): 0, 0.1, 0.3, 0.5, 1.0, 1.5, 2.0, 2.5 and 3.0. Figure 3.4 dipicts the specimen, in which an EDM crack is located in the middle of the 4140-steel block.

The Bi-slide linear positioning system allows automated and accurate positioning of the ultrasonic transducer on the specimen's surface. It is controlled by Windows-based GUI called COSMOS. It is programmed to linearly move the transducer a distance of 30mm with a step size of 0.25mm. At every step, a pulse is sent to the OmniScan unit to mark the current position of the transducer. The starting position

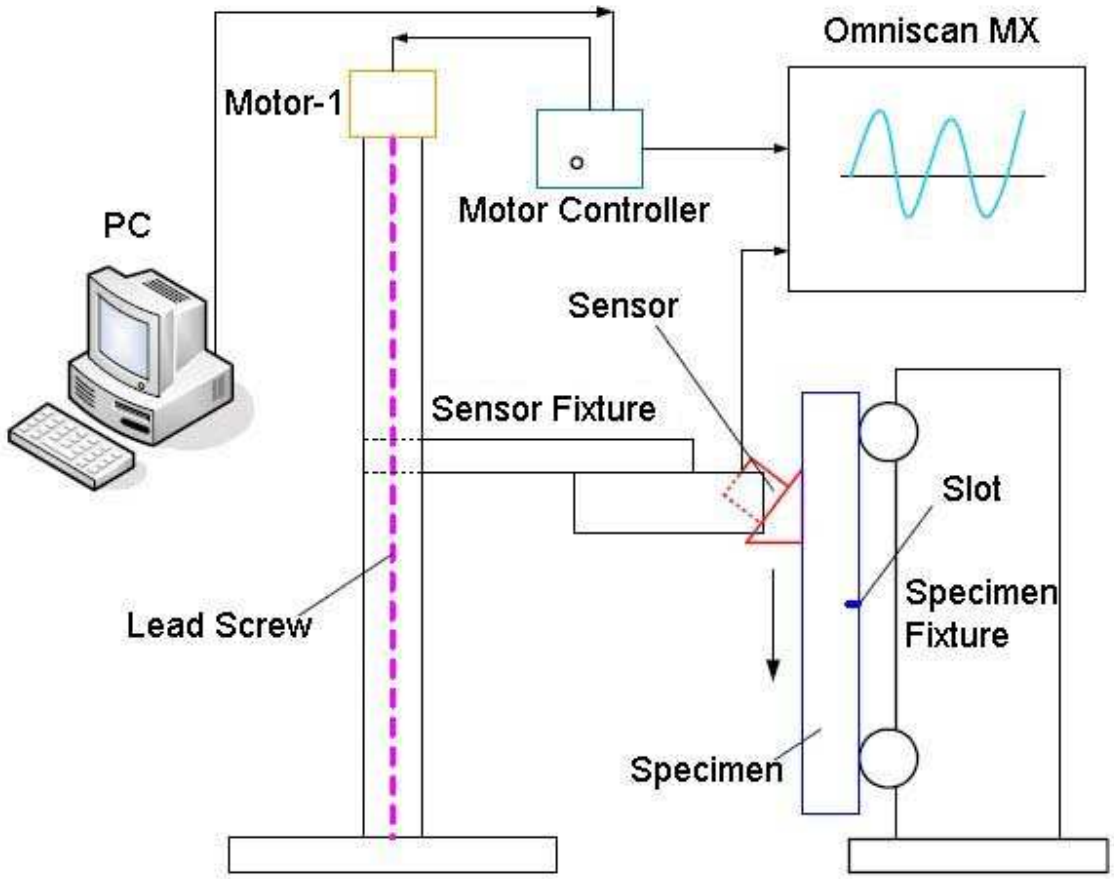


Figure 3.3: Schematic of the complete experimental system [36]

of the transducer is set at approximately 30mm measured from the probe beam index to the location of the EDM slot.

At each step of the Bi-slide movement, an ultrasonic pulse is generated and transmitted to the specimen. The reflected ultrasonic echo is recorded by the OmniScan unit over 2048 data points. For a travel distance of 30mm with a step size of 0.25mm, there are 121 positions for ultrasonic data collection. Table 3.1 shows the parameters which were selected on the OmniScan unit.

The gain levels used for each specimen were from 5.0dB to 60.0dB with an incremental size of 2.5dB. There were a total of 23 different gain levels. When a gain

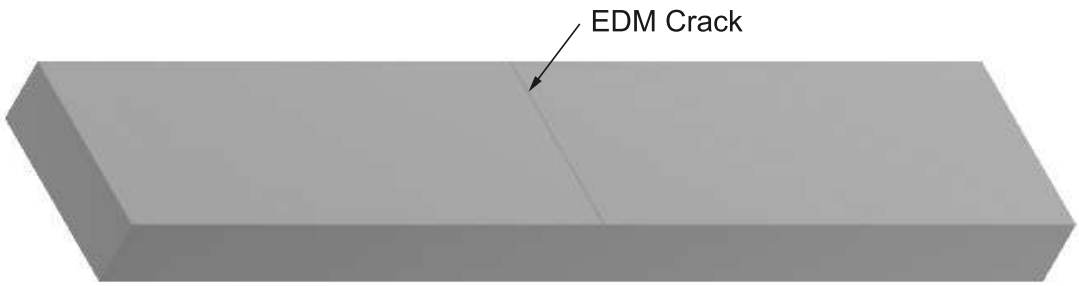


Figure 3.4: An example of an EDM crack specimen

level was fixed, three separate scans were performed on the same specimen.

3.1.3 The A-Scan and B-Scan Data Sets

Using the experimental system shown in Figure 3.3, we collected $9 \times 23 \times 3$ ultrasonic data files, where "9" represents the nine specimens with different slot depths; "23", the different gain levels for each specimen with values ranging from 5dB to 60dB and an incremental size of 2.5dB; and "3", the number of data sets collected under identical testing conditions (namely identical specimens and identical gain levels). Each file contains 121×2048 data points, where "121" denotes the number of sensor positions from which A-scan waveforms were collected and "2048", the number of data points at each sensor location.

Figure 3.5 illustrates signals obtained at the gain levels of 10dB, 40dB and 60dB, for the specimen with a 10mm crack. Each has three axes represented by time, position and echo. At a gain level of 60dB (the picture on the right in Figure 3.5), some signals are cut off. Actually, for these nine kinds of specimen, if the gain level is equal or less than 50dB, the cutting-off of the signal can be avoided. If a signal is

Table 3.1: Parameters selected on the OmniScan unit

Parameter Name	Parameter Value	Parameter Description
Wedge Delay	$5.83\mu s$	Time for wave traveling through the transducer and the wedge
Material sound velocity	3329.4 m/s	Velocity of wave inside the specimen
Range	23.38mm	Length of X-axis on the display of OmniScan
Data sample size	2048 points	
Scanning distance	30mm with 0.25mm/step	We have 121 steps
Input Voltage	200V	Maximum value of the input signal
Pulse width	35ns	the width of the input pulse
PRF	Optimum	Pulse repetition frequency
Filter	None	
Rectification Mode	RF (Radio Frequency)	No rectification is performed
Scale	1	Scaling of the maximum sampling frequency

cut off, the features extracted in later studies will be affected. For example, we want to calculate the time between the 25% level and the peak; if the peak is cut off, the value given time doesn't reflect the real situation. Because different gain levels with values from 5dB to 50dB and an incremental size 2.5dB are studied in this paper, 19 different gain levels have been used instead of 23 (5dB∼60dB).

Figure 3.6 shows the data points for the specimen with a 1.0mm crack at gain level 40dB. Figure 3.6(a) represents the data in B-Scan form and Figure 3.6(b) represents the data in A-Scan form. In Figure 3.6(a), the X-axis shows the time points varying from $1 \sim 2048$, and the Y-axis shows the transducer's position varying from $1 \sim 121$. In Figure 3.6(b), the X-axis shows the time points varying from $1 \sim 2048$, and the Y-axis shows the echo's amplitude.

To find good features for crack classification, we first employ two data prepro-

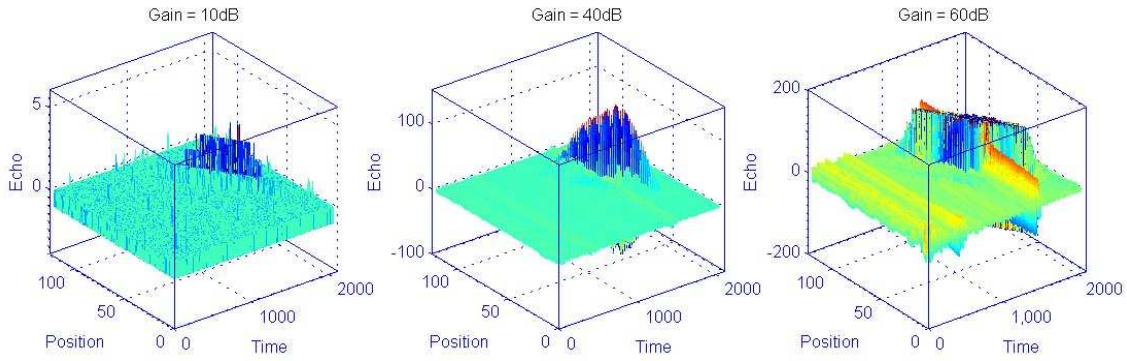
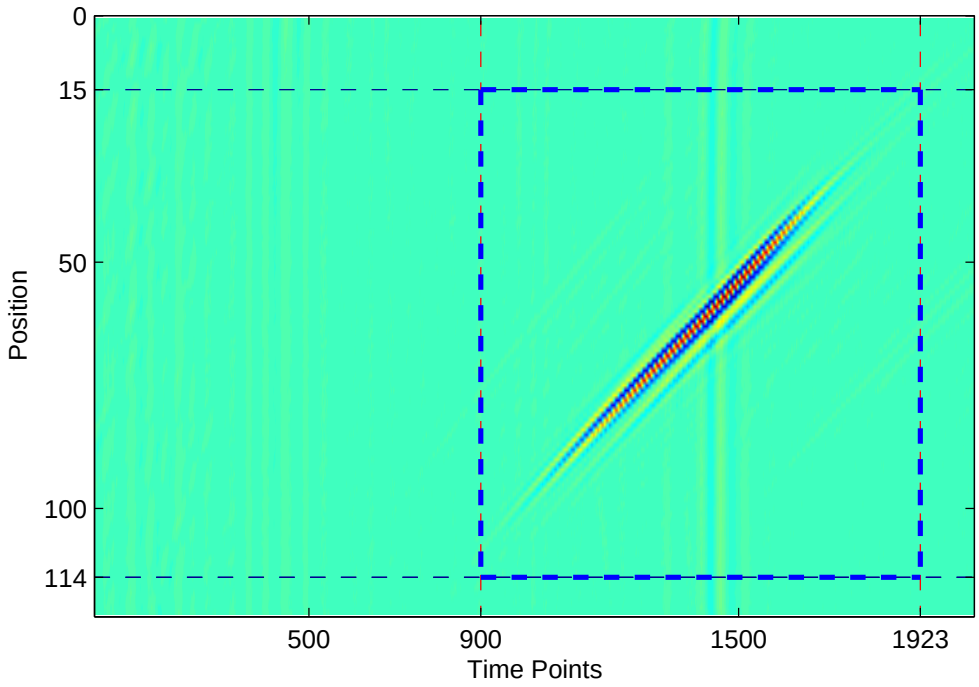


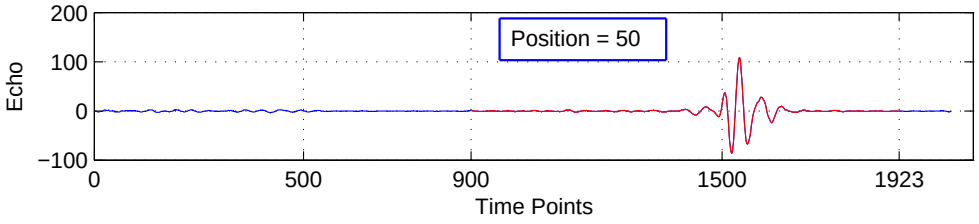
Figure 3.5: Signals obtained at different gain levels for the specimen with a 1.0mm crack

cessing techniques called averaging and gating [28]. The three data files are averaged under identical testing conditions to form one data file. To eliminate the signals caused by the wedge and specimen interface, a passing gate is set between the time points from 900 to 1923. In Figure 3.6(a), from the time point 900, an echo will bounce off the crack. In the range between the time points 900 to 1923, all crack echos will be covered. Notes as well, that the value of the time points total, which is 1024 $(1923 - 900 + 1)$, or 2^{10}, is convenient for using fast fourier transform. To avoid spurious effects at the beginning and end of the bi-slide movement, we have further removed the first 14 and the last 7 scanning positions (leaving 100 positions remaining) [37]. Thus, the data points in Figure 3.6(a), those surrounded by the dashed box, are the ones that will be studied in this research work. Each datum has $1024 \times 100 = 102400$ data points. In Figure 3.6(b), the red part of the data, which corresponds to the data in position 50 of Figure 3.6(a), are the data points we will study. Each set of data has 1024 data points. For convenience, we will call the data in Figure 3.6(a) the **B-Scan data set** and the data in Figure 3.6(b) the **A-Scan data set**. Totally, there are 171 (9×19) B-Scan data sets and 17100 $(9 \times 19 \times 100)$ A-Scan data sets, where "9" denotes the nine specimens "19" denotes the different gain

levels for each specimen with values ranging from 5dB to 50dB with an incremental size of 2.5dB, and "100" denotes the bi-slide movement position.



(a) B-Scan form



(b) A-Scan form

Figure 3.6: Data points for a specimen with a 1.0mm crack at gain level 40dB

This Section has defined and trimmed the A-Scan and B-Scan data sets. The following section further process the A-Scan data sets for feature extraction.

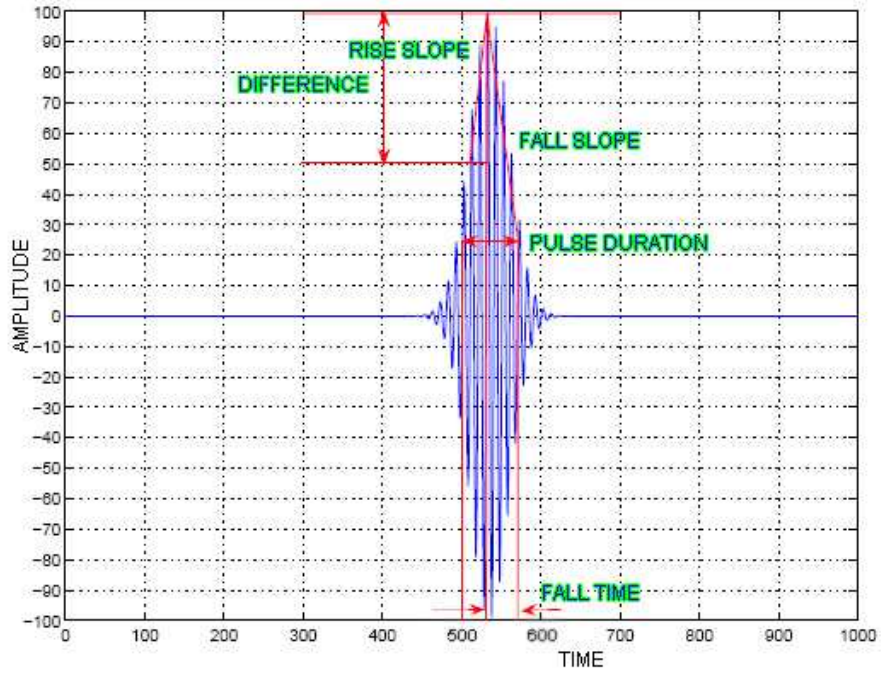
3.2 Feature Extraction from A-Scan Data Set

The A-Scan and B-Scan data sets, obtained are not used directly as input in the SVM classifier. Features, the individual measurable properties of the data being collected, need to be extracted first from those data. Choosing discriminating and independent features is a key step to success with the classification algorithm [32].

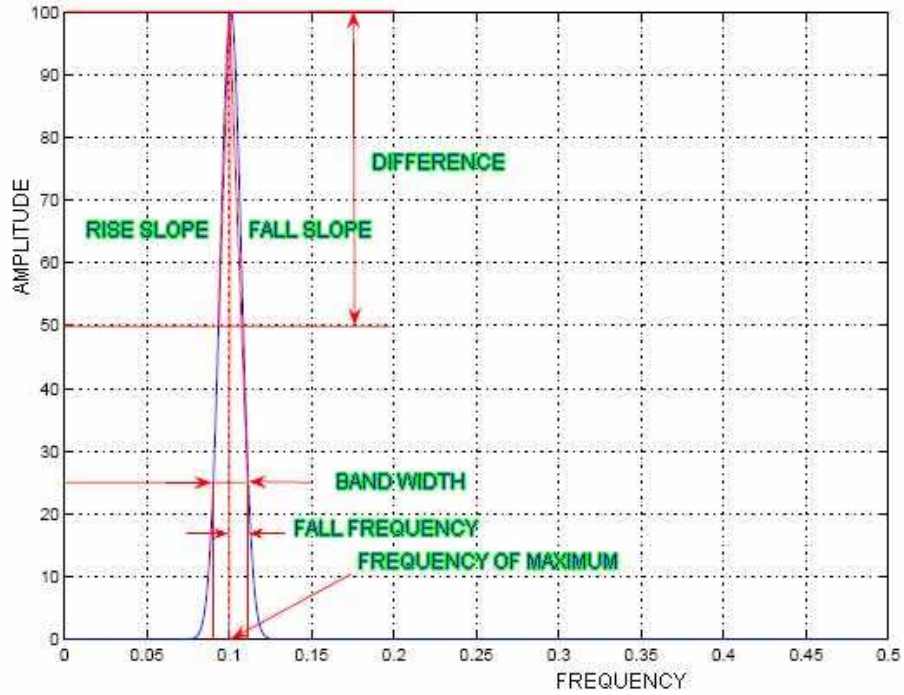
Features play an important role in determining pipe crack sizes by means of ultrasonic tests. Appropriate features improve classification performance. Features can be extracted from ultrasonic data by using the statistics of waveform or signal processing technologies such as Fourier transform and wavelet analysis.

3.2.1 Available Features from the Time and Frequency Domains

In the time domain, certain features have been used to describe an ultrasonic signal or pulse. Among these are the statistics of waveform amplitude, pulse duration, and local and global rise and fall indexes. By means of Fourier Transform, a signal in the time domain can be transformed into the frequency domain; statistics of the obtained power spectrum, such as structural parameters and local and global rise and fall relative indexes, have been used to characterize an ultrasonic signal in the frequency domain [38][6]. Features characterizing an ultrasonic signal in the time and frequency domains are illustrated in Figures 3.7(a) and 3.7(b), respectively [31]. Reference [38] lists 69 time and frequency domain features for flaw classification of pipes. These are shown in Table 3.2.



(a) Features in the time domain



(b) Features in the frequency domain

Figure 3.7: Features in the time and frequency domains

Table 3.2: Features in the time and frequency domains for flaw classification of pipelines [38]

RF: radio frequency, CD: cumulative distribution

No	feature		
	definition	domain	description
1	mean value of the normalized RF waveform amplitude values	time	statistics
2	variance of the normalized RF waveform amplitude values		
3	mean value of the normalized envelope function		
4	variance of the normalized envelope function		
5	difference between 50% level and 25% level (RF waveform CD)		diference
6	difference between 75% level and 25% level (RF waveform CD)		
7	difference between 90% level and 25% level (RF waveform CD)		
8	local pulse duration between 25% levels		pulse
9	global pulse duration between 25% levels		
10	local rise time from 25% level to peak		local rise
11	local rise time from 50% level to peak		
12	local rise variance between 25% level and peak		
13	local rise variance between 50% level and peak		
14	local rise slope between 25% level and peak		
15	local rise slope between 50% level and peak		
16	local fall time from peak to 25% level		local fall
17	local fall time from peak to 50% level		
18	local fall variance between 25% level and peak		
19	local fall variance between 50% level and peak		
20	local fall slope between 25% level and peak		
21	local fall slope between 50% level and peak		
22	global rise time from 25% level to peak		global rise

To Be Continued

Continued

No	feature		
	definition	domain	description
23	global rise time from 50% level to peak		
24	global rise variance between 25% level and peak		
25	global rise variance between 50% level and peak		
26	global rise slope between 25% level and peak		
27	global rise slope between 50% level and peak		
28	global fall time from peak to 25% level		global fall
29	global fall time from peak to 50% level		
30	global fall variance between 25% level and peak		
31	global fall variance between 50% level and peak		
32	global fall slope between 25% level and peak		
33	global fall slope between 50% level and peak		
34	frequency of the maximum value of the power spectrum	frequency	structural
35	center frequency of the power spectrum		parameters
36	measured bandwidth		
37	mean value of the normalized power spectrum		statistics
38	variance of the normalized power spectrum		
39	difference between 50% level and 25% level (spectrum CD)		difference
40	difference between 75% level and 25% level (spectrum CD)		
41	difference between 90% level and 25% level (spectrum CD)		
42	fraction of total power between lower 25% level and peak		fraction
43	fraction of total power between lower 50% level and peak		
44	fraction of total power between peak and upper 25% level		
45	fraction of total power between peak and upper 50% level		
46	local rise frequency from 25% level to peak		local rise
47	local rise frequency from 50% level to peak		
48	local rise variance between 25% level and peak of spectrum		

To Be Continued

Continued

No	feature		
	definition	domain	description
49	local rise variance between 50% level and peak of spectrum		
50	local rise slope between 25% level and peak of spectrum		
51	local rise slope between 50% level and peak of spectrum		
52	local fall frequency from peak to 25% level		local fall
53	local fall frequency from peak to 50% level		
54	local fall variance between peak of spectrum and 25% level		
55	local fall variance between peak of spectrum and 50% level		
56	local fall slope between peak of spectrum and 25% level		
57	local fall slope between peak of spectrum and 50% level		
58	global rise frequency between 25% level and peak of spectrum		global rise
59	global rise frequency between 50% level and peak of spectrum		
60	global rise variance between 25% level and peak of spectrum		
61	global rise variance between 50% level and peak of spectrum		
62	global rise slope between 25% level and peak of spectrum		
63	global rise slope between 50% level and peak of spectrum		
64	global fall frequency between peak of spectrum and 25% level		global fall
65	global fall frequency between peak of spectrum and 50% level		
66	global fall variance between peak of spectrum and 25% level		
67	global fall variance between peak of spectrum and 50% level		
68	global fall slope between peak of spectrum and 25% level		
69	global fall slope between peak of spectrum and 50% level		

In paper [39], Osterried *et al.* [39] use skewness as a signature for studying echoes. Skewness is defined as

$$\text{Skewness} = \frac{E(\boldsymbol{X} - \mu)^3}{\sigma^3}$$

where $\boldsymbol{X}$ is the input vector, μ is the mean of $\boldsymbol{X}$, σ is the standard deviation of $\boldsymbol{X}$, and $E(\cdot)$ represents the expected value, $(\boldsymbol{X} - \mu)^3$. Skewness is a measure of the asymmetry of the data around the sample mean. If the skewness is negative, the data are spread out more to the left of the mean than to the right. If the skewness is positive, the data are spread out more to the right [40]. In paper [41], Lei *et al.* show that the kurtosis measured in the time and the frequency domain are good predictors of the relative magnitude and frequency distribution of the acoustic trauma (hearing loss). Kurtosis is defined as

$$\text{Kurtosis} = \frac{E(\boldsymbol{X} - \mu)^4}{\sigma^4}$$

where $\boldsymbol{X}$ is the input vector, μ is the mean of $\boldsymbol{X}$, σ is the standard deviation of $\boldsymbol{X}$, and $E(\cdot)$ represents the expected value, $(\boldsymbol{X} - \mu)^4$. Kurtosis is a measure of how outlier-prone a distribution is. The kurtosis of a normal distribution is 3. Distributions that are more outlier-prone than normal have a kurtosis value of greater than 3; distributions that are less outlier-prone have a value of less than 3 [40].

The features listed above are obtained from the time domain or the frequency domain, separately. Features extracted from the time-frequency joint domain are also useful in flaw identification, and certain time-frequency analysis methods have been applied to extracting features from ultrasonic signals [42][43][44]. Discrete Wavelet Transform (DWT) is an effective one among those various time-frequency analysis methods [31].

3.2.2 The Features to be Used

Based on Section 3.2.1, we list many features in the time, frequency and time-frequency joint domains that can be extracted from ultrasonic data. In fact, there are far more than that. For the sake of simplicity, we will consider only 14 features which were extracted from A-Scan data set. Those features are listed in Table 3.3. In this table, features #1, #2, #4, #5, #6, #7, #10, #11 and #12 are based on paper [38]. Features #3, #8 and #9 are concepts for identifying digital signals. Features #13 and #14 are based on papers [39] and [41], respectively. We believe these features adequately cover different categories such as statistics, pulse, rise and fall. We have not, however, focused on studying all the kinds of features that can characterize the data of ultrasound echoes. The late work is based only on the limited features in Table 3.3.

Table 3.3: Features Extracted from Ultrasonic Data

Feature#	**Description**	**Definition**
F1	Mean of RF waveform	
F2	Variance of RF waveform	
F3	Energy in time domain	Energy $= \int \|s(t)\|^2 dt$
F4	Mean of Envelope	mean of $\|s(t) + j\widehat{s}(t)\|$
F5	Variance of Envelope	variance of $\|s(t) + j\widehat{s}(t)\|$
F6	Rising time from 25% level to peak	
F7	Falling time from peak to 25% level	
F8	Time center	$t_0 = \frac{1}{Energy} \int t\|s(t)\|^2 dt$
F9	Pulse duration	$\frac{1}{Energy} \int (t - t_0)^2 \|s(t)\|^2 dt$
F10	Peak frequency	

F11	Center frequency	$\Omega_0 = \frac{1}{Energy}\int_{-\infty}^{\infty}\Omega\|S(\Omega)\|^2 d\Omega$
F12	Band width	$B = \frac{\pi}{Energy}\int_{-\infty}^{\infty}(\Omega-\Omega_0)\|S(\Omega)\|^2 d\Omega$
F13	Skewness of magnitude of spectrum	Skewness of $\|S(\Omega)\|$
F14	Kurtosis of magnitude of spectrum	Kurtosis of $\|S(\Omega)\|$

Note:

1. Features #1∼#9 are in the time domain. Features #10∼#14 are in the frequency domain.
2. $s(t)$ denotes the observed signal, $\widehat{s}(t)$ the Hilbert transform of $s(t)$, and $S(\Omega)$ the Fourier transform of $s(t)$. *Energy* is the value of F3.

For each A-Scan data set, we can get the 14 features as listed in Table 3.3. To form an input vector for classification, we treat each feature as one dimension in the input vector. The input vector needs 14 dimensions to represent these 14 features. Also, each A-Scan data set corresponds to a Gain value (5dB $\sim$ 50dB) and a position value (15 $\sim$ 114), as illustrated in Figure 3.6. We can use two other dimensions in the input vector to denote the Gain and position values, given the input vector a total of 16 dimensions. We will call this input vector the **A-Scan feature data set**. The number of A-Scan feature data sets are 17100, equal to the number of A-Scan data sets. Just as we have both an A-Scan data set and a B-Scan data set, we can also form another input vector called a **B-Scan feature data set**. Since there are 100 A-Scan data sets in each B-Scan data set, from which 14 features are extracted, we get 1400 features in a B-Scan data set. Also one B-Scan data set corresponds to a Gain value, which needs one dimension of storage. A B-Scan feature data set has 1401 dimensions, and the number of B-Scan feature data sets is 171, which is the number of B-Scan data sets.

Either A-Scan feature data sets or B-Scan feature data sets can be used as the

input vectors for the classification. The following will discuss which input vector is better when the test error for the classification is taken into account. Due to limited time, only the better one will be studied further.

3.2.3 Discussion of A-Scan Feature Data Sets and B-Scan Feature Data Sets

Since in the experiment, we have nine kinds of specimens with slot depths varying from 0mm to 3mm, a binary classification problem require us to address 8 kinds of cases (see Table 3.4). For example, Case #4 represents a classification that separates the input vector (either A-Scan or B-Scan feature data sets) into two classes. When the crack size is $\leq 0.5mm$, we classify those data as $+1$ class. On the other hand, when the crack size is $> 0.5mm$, we classify those data as -1 class.

Table 3.4: List of separation definitions

Case	Separation Criteria
Case #1	$\{y = +1 \mid crack\ size = 0mm\}\ vs.\ \{y = -1 \mid crack\ size > 0mm\}$
Case #2	$\{y = +1 \mid crack\ size \leq 0.1mm\}\ vs.\ \{y = -1 \mid crack\ size > 0.1mm\}$
Case #3	$\{y = +1 \mid crack\ size \leq 0.3mm\}\ vs.\ \{y = -1 \mid crack\ size > 0.3mm\}$
Case #4	$\{y = +1 \mid crack\ size \leq 0.5mm\}\ vs.\ \{y = -1 \mid crack\ size > 0.5mm\}$
Case #5	$\{y = +1 \mid crack\ size \leq 1.0mm\}\ vs.\ \{y = -1 \mid crack\ size > 1.0mm\}$
Case #6	$\{y = +1 \mid crack\ size \leq 1.5mm\}\ vs.\ \{y = -1 \mid crack\ size > 1.5mm\}$
Case #7	$\{y = +1 \mid crack\ size \leq 2.0mm\}\ vs.\ \{y = -1 \mid crack\ size > 2.0mm\}$
Case #8	$\{y = +1 \mid crack\ size \leq 2.5mm\}\ vs.\ \{y = -1 \mid crack\ size > 2.5mm\}$

This experiment uses the SVM and Kernel Methods Matlab Toolbox[1] written

[1]http://asi.insa-rouen.fr/enseignants/ arakotom/toolbox/index.html

by Canu *et al.* for SVM classification. For each input vector, we use the Gaussian kernel as the kernel function. The parameters (C, σ) of SVM are selected by the grid-search method [45]. The optimal (C, σ) for A-Scan feature data sets are $C = 50$ and $\sigma = 2$. And the optimal (C, σ) for B-Scan feature data sets are $C =$ infinity and $\sigma = 1000$. We will further discuss the grid search method later in the Chapter 4. For each case, we employ the 5-folded cross validation [46] to determine the test error. In 5-fold cross-validation, the original input vectors are partitioned into 5 subsets. Of the 5 subsets, a single subset is retained as the validation data for testing, and the remaining 4 subsets are used as training data. The cross-validation process is then repeated 5 times (the folds), with each of the 5 subsets used exactly once as the validation data. The 5 results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method is that all input vectors are used for both training and validation, and each input vector is used for validation exactly once [32]. Table 3.5 shows the classification test error for these eight cases, using the A-Scan and B-Scan feature data sets as the input vectors separately.

From Table 3.5, B-Scan feature data sets have a lower test error than do A-Scan feature data sets. Except for Case #2, the test error with B-Scan feature data sets is smaller than that with A-Scan feature data sets. For the remaining parts of this research, we will focus solely on the B-Scan feature data sets.

In summary, 14 features in the time and frequency domains are extracted and formed into A-Scan feature data sets and B-Scan feature data sets. A comparable test is also done using the A-Scan and B-Scan feature data sets as the input vectors for the classification. In the following section, further feature reduction is studied in order to find the best dimension subsets of the B-Scan feature data sets.

Table 3.5: Test error for A-Scan feature data sets and B-Scan feature data sets

Case#	A-Scan feature data sets	B-Scan feature data sets
Case #1	7.72%	3.92%
Case #2	4.74%	5.85%
Case #3	6.49%	0
Case #4	7.19%	2.92%
Case #5	6.67%	2.92%
Case #6	7.89%	4.68%
Case #7	8.42%	3.51%
Case #8	8.07%	4.09%

3.3 Further Feature Reduction for B-Scan Feature Data Sets

3.3.1 Introduction of Feature Reduction

For each B-Scan feature data set, there are 1401 dimensions which consist of 14 kinds of extracted features (100 of each kind) and the Gain value. Based on these data sets, we want to find a subset of the dimensions. An efficient subset, which not only reduces the dimensions of the feature set but also improves the classifier's performance, needs to be extracted [47].

Intuitively, the optimal subset can be determined by doing an exhaustive search in the feature set. But an exhaustive search is feasible for only a small number of features (or dimensions). When the features become larger, the explosive computational cost makes the exhaustive search impractical. In paper [48], Backer *et al.* proposed the Max-min method which is computationally efficient. It evaluates only

the individual and pairwise merits of features. Siedlecki *et al.* introduced the use of the genetic algorithm (GA) for this problem and obtained good results [49][50], but it is necessary to to carefully select and test the parameters in GA [47]. In paper [51], the author derives orthogonal forward selection (OFS) and orthogonal backward elimination (OBE) algorithms for feature subset selection by incorporating Gram−Schmidt and Givens' orthogonal transforms into forward selection and backward elimination procedures, respectively. Zhang *et al.* [47] use tabu search to select an optimal subset from the original large feature set for use as a pattern classifier. In this work, we will use the sequential backward selection method (SBS) and sequential forward selection method (SFS) (to be defined next) which are widely used for subset selection [13].

3.3.2 Feature Reduction with SBS and SFS

In **sequential backward selection (SBS)**, we start with all the features and delete one at a time, the one whose removal cause the least deterioration in the selection criterion is removed first. Sometimes it is possible to delete more than one feature at a time on the basis of feature ranking. This is done until some selection criterion is met, such as obtaining the lowest test error. In **sequential forward selection (SFS)**, features are sequentially added to an empty candidate set until the addition of further features does not decrease the test error.

Compared with exhaustive search, which has $2^n - 1$ non-empty subsets of an $n-$feature data set and is typically infeasible (depending on the size of n and the cost of objective calls), sequential searches move in only one direction, always growing or always shrinking the candidate set. In paper [47], Zhang *et al.* compared the computation cost for different kinds of feature reduction algorithms for a 30-dimension data set and a 100-dimension data set. These algorithms include SFS,

SBS, Generalized SFS, Generalized SBS, as well as l take away r methods (PTA), sequential forward floating selection (SFFS), sequential backward floating selection (SBFS) and GA [49][13][48][52]. The computation cost of the SFS or SBS algorithms is much lower than that of other algorithms.

Neither SBS nor SFS examines all possible feature subsets, so it cannot be guaranteed either will produce the optimal result. In SBS, the features discarded cannot be re-selected. In SFS, the features selected cannot be removed later. To overcome this, we will combine these two methods to do the feature reduction.

Usually, sequential backward selection is slower but rather is more stable in selecting optimal features than sequential forward selection [13]. For this reason we will use SBS first and then do the SFS. From the SBS, we get an optimal subset of features, on the basis of which the SFS can be done. In this procedure, the features discarded in the SBS process can be chosen again.

3.3.3 A Feature Reduction Experiment for B-Scan Feature Data Sets

Before doing the experiment for feature reduction, we need to prepare the experiment's data. Table 3.6 lists descriptions of the experiment's data. As we mentioned above, we will use B-Scan feature data sets as the input vector, these consist of a total of 171 observations. If we want to separate these data into two classes, i.e., a large crack size class and a small crack size class, we could have 8 kinds of separation which corresponds to Case #1 $\sim$ Case #8 (see Table 3.4). The output can take only the values $\{\pm 1\}$ which label the large and small crack sizes respectively. For each separation, by partitioning the data into training data (114 observations, 2/3 of the total) and test data (57 observations, 1/3 of the total), we randomly generate

20 groups of data sets (sets of training and test data), and label these #1, #2, $\cdots$, #20.

Table 3.6: SBS and SFS experiment data set descriptions

Separation Criteria	Case #1 $\sim$ Case #8, see Table 3.4
Number of Training data	114
Number of Test data	57
Dimensions of Input Vectors	1401 (B-Scan feature data set)
Associate Labels	± 1
Number of Groups	20

Figure 3.8 illustrates the steps for doing the sequential forward selection and sequential backward selection. After we prepare 20 groups of training data and test data for each case, we use them as input for the binary SVM classifier with Gaussian kernel and obtain the test errors. For the SVM classifier, parameters C = infinity and $\sigma = 1000$ are chosen by the grid search method as being optimal for considering the sum of average errors for all the Cases (Cases #1 $\sim$ #8). To simplify the test, we will use these parameters for all the SBS and SFS tests. Now we can average the test errors of the 20 groups of test data for each case; the sum of these 8 average test errors, or E_p, are shown in Figure 3.8. Table 3.7 shows the average test error for each case in the full dimensional B-Scan feature data sets. The sum of the average errors is 27.89%.

That done, we begin to do the SBS process. That is to say, from a full dimensional B-Scan feature data set, we reduce the dimensions for both the training data and the test data. Eventually, we find a subset which has the lowest sum of average test errors. For the convenience of discussion, we call the features listed in Table 3.3 numbered features. Then each B-Scan feature data set (1401 dimensions) consists

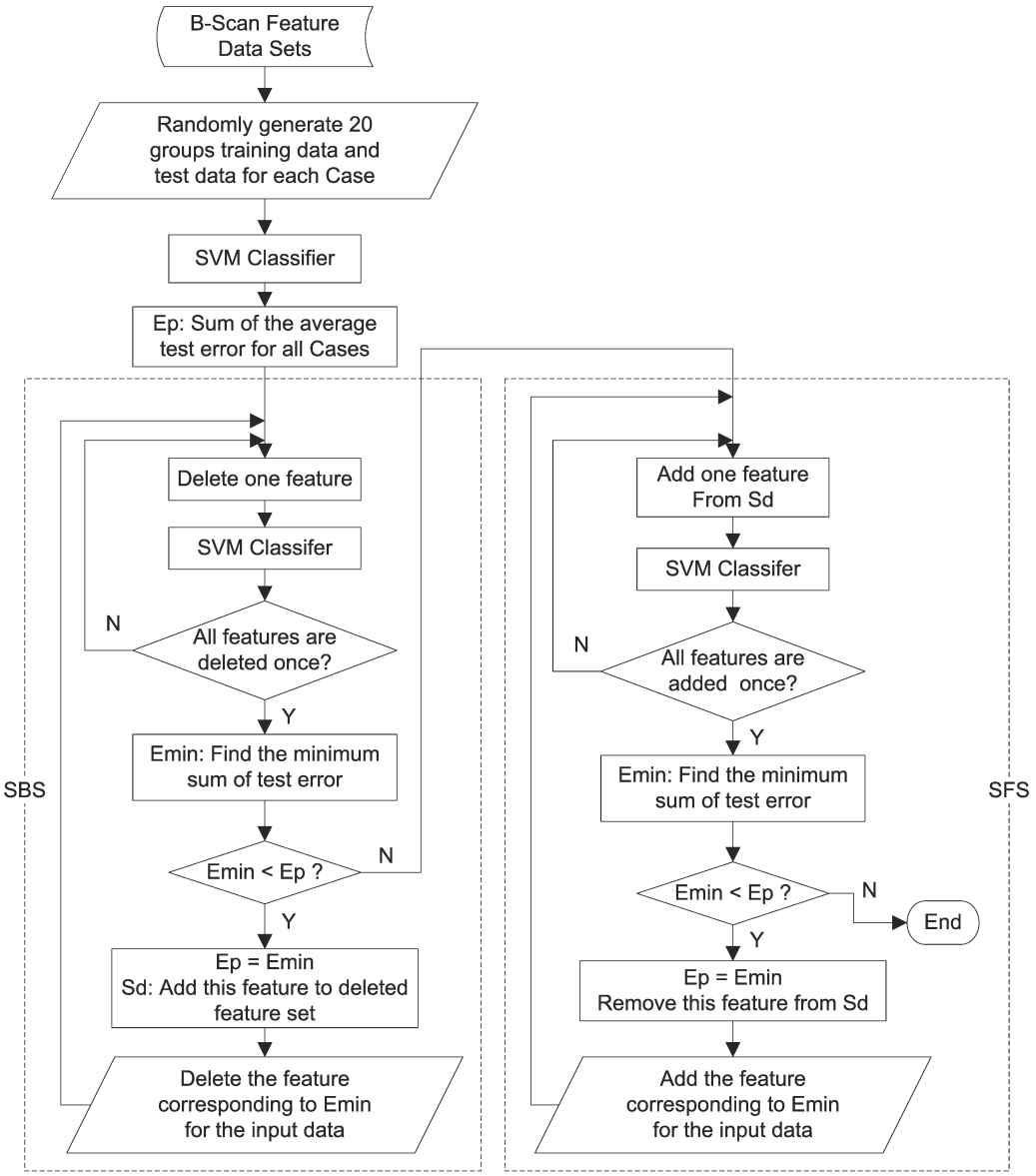


Figure 3.8: Flow chart of SBS/SFS process

Table 3.7: Average test error of B-Scan feature data sets

	Average test error
Case #1	7.15%
Case #2	5.17%
Case #3	0.17%
Case #4	2.31%
Case #5	3.00%
Case #6	5.80%
Case #7	5.33%
Case #8	3.52%
Sum	27.89%

of 100 groups of numbered features represented by 14×100 dimensions and one dimension of gain value. In other words, we have a total of 15 kinds of features. For the classification, assume those dimensions with the same numbered feature have the same effect. In SBS/SFS, we may delete/add 100 dimensions at one time except for the dimension of Gain. Thus in Figure 3.8, when we say "delete/add one feature", this means either deleting/adding 100 dimensions of the numbered features or deleting/adding the Gain value. Also the feature in Figure 3.8 refers to one kind of feature.

Tables 3.8 $\sim$ 3.18 show the results of SBS process. To illustrate this (see Figure 3.8 and the results in the tables), we list the steps for the SBS.

Step 1 Delete one kind of feature for the training data and test data, compute the test errors and average them, then record the average test errors in a table.

Step 2 Repeat **Step 1** until all types of feature have been deleted exactly once.

Step 3 Sum the average test errors for each column in the table and find the minimum one, named E_{min} in Figure 3.8.

Step 4 If $E_{min} < E_p$ (E_p is a intermediate variable used to record the previous value of E_{min})

Let $E_p = E_{min}$ and insert the feature number corresponding to the minimum sum into a set, S_d, for use by the SFS process. Delete the feature corresponding to E_{min} from the input data and repeat **Step 1**.

If $E_{min} \geq E_p$

Stop the SBS process and begin the SFS process

In Table 3.8, when feature #11 is deleted, a minimum sum of average test errors ($E_{min} = 24.3\%$) is obtained. Since $E_{min} < E_p = 27.89\%$ (in Table 3.7), a second iteration is needed. As $E_{min} = 13.9\%$ (Table 3.18) is greater than $E_p = 12.7\%$ (Table 3.17), the SBS process is stopped. After 11 iterations, five features (four numbered features and 1 gain feature) are kept. The 5 features selected by the SBS method (see in Table 3.18) are:

- Feature #4, mean of envelope,
- Feature #9, pulse duration,
- Feature #12, band width,
- Feature #13, skewness of magnitude of spectrum, and
- Gain

Using the result of the SBS, we now use SFS to see if any other features can be added to the group of features obtained. Similarly, SFS takes several steps.

Step 1 add one kind of feature to set S_d for the training data and test data, compute the test errors and average them, then record the average test errors in a table.

Step 2 Repeat **Step 1** until all types of feature have been added exactly once.

Step 3 Sum those average test errors for each column in the table and find the minimum one, named E_{min}.

Step 4 If $E_{min} < E_p$

Let $E_p = E_{min}$ and remove the feature number corresponding to the minimum sum from set S_d. Add the feature corresponding to E_{min} for the input data and repeat **Step 1**.

If $E_{min} \geq E_p$

Stop the SFS process. The SBS/SFS process is finished.

Table 3.19 shows the 1^{st} iteration of sequential forward selection after the SBS. As the minimum sum value (E_{min}) of 13.2%, is greater than the E_p value of 12.7%, the SFS process is stopped.

The SFS process shows that adding any one of the other features to those five features can not lower the test error any more. Thus the subset of the features combining those five kinds of features is the optimal one for classification. It should be pointed out that if we want to use the SBS/SFS method to find the optimal subset for any particular cases, say Case #1, the selection criteria should use the average error for that case, Case #1, instead of the sum of the average test errors.

Table 3.8: First iteration of SBS

Case	Average Error Rates for Feature Deleted(%)														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Gain
#1	6.5	6.6	6.6	6.9	6.8	6.3	7.7	9.7	6.8	7.2	6.8	7.0	6.8	6.6	6.9
#2	5.4	5.3	5.3	5.3	5.3	4.6	4.9	6.6	5.3	5.2	2.7	5.3	5.4	5.5	5.2
#3	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.1	0.2	0.2	0.2	0.2
#4	2.5	2.3	2.3	2.3	2.3	2.3	1.9	2.3	2.2	1.6	1.0	2.4	2.1	2.2	2.4
#5	3.3	3.1	3.1	3.1	3.1	3.0	3.4	4.5	3.2	2.8	2.3	3.2	3.2	3.4	3.2
#6	6.1	5.8	5.8	5.8	5.8	6.4	6.8	6.8	6.0	5.9	3.8	6.0	6.2	6.1	6.0
#7	4.7	5.2	5.2	5.2	5.2	4.4	5.7	5.8	5.3	5.3	4.9	5.2	5.5	5.4	5.2
#8	4.0	3.8	3.8	3.8	3.8	4.1	3.1	4.3	3.8	3.9	2.7	3.9	3.9	4.0	3.8
Sum	32.6	32.3	32.3	32.5	32.4	31.3	33.7	40.2	32.8	32.0	24.3	33.0	33.3	33.2	32.8

Action: Delete feature (#11)

Table 3.9: Second iteration of SBS

Case	Average Error Rates for Feature Deleted(%)													
	1	2	3	4	5	6	7	8	9	10	12	13	14	Gain
#1	5.8	6.7	6.7	6.9	6.7	6.3	7.7	7.6	7.0	6.5	6.8	6.8	7.0	6.7
#2	2.5	2.7	2.7	2.7	2.7	2.0	2.3	3.6	2.9	2.8	2.7	2.8	2.8	2.8
#3	0.2	0.1	0.1	0.1	0.1	0.1	0.0	0.1	0.1	0.1	0.1	0.1	0.1	0.1
#4	1.6	1.0	1.0	1.0	1.0	1.5	0.6	1.1	1.0	1.0	1.1	1.0	1.2	1.0
#5	2.6	2.3	2.3	2.3	2.3	2.2	1.4	3.5	2.3	2.2	2.3	2.3	2.3	2.3
#6	3.2	3.8	3.8	3.8	3.8	3.9	3.4	4.6	3.9	3.9	3.8	4.0	4.2	3.8
#7	4.2	4.7	4.7	4.9	4.8	4.4	5.0	6.4	5.3	4.6	5.1	5.6	5.5	4.9
#8	3.0	2.6	2.6	2.7	2.6	2.5	2.8	3.7	2.8	3.3	2.7	2.8	2.8	2.7
Sum	23.1	23.9	23.9	24.4	24.0	22.8	23.2	30.6	25.3	24.3	24.5	25.4	25.9	24.3

Action: Delete feature (#6)

Table 3.10: Third iteration of SBS

Case	Average Error Rates for Feature Deleted(%)												
	1	2	3	4	5	7	8	9	10	12	13	14	Gain
#1	5.2	6.3	6.3	6.8	6.6	8.0	9.3	6.9	6.6	6.6	6.6	6.4	6.2
#2	2.4	2.0	2.0	2.0	2.0	1.2	2.9	2.4	2.3	2.1	2.0	2.2	2.2
#3	0.2	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1
#4	2.1	1.5	1.5	1.6	1.5	0.3	2.1	1.6	1.4	1.6	1.6	1.6	1.5
#5	2.3	2.2	2.2	2.2	2.2	0.2	3.4	2.3	2.1	2.2	2.2	2.2	2.2
#6	3.8	3.9	3.9	3.9	3.9	2.8	4.7	4.0	3.8	3.9	3.9	4.0	3.9
#7	3.5	4.4	4.4	4.2	4.4	3.7	4.6	4.5	3.5	4.5	4.6	4.6	4.4
#8	2.8	2.5	2.5	2.5	2.5	2.4	3.5	2.5	2.8	2.5	2.6	2.9	2.5
Sum	22.2	22.8	22.8	23.3	23.2	19.4	30.7	24.3	22.7	23.4	23.6	23.9	22.9

Action: Delete feature (#7)

Table 3.11: Fourth iteration of SBS

Case	Average Error Rates for Feature Deleted(%)											
	1	2	3	4	5	8	9	10	12	13	14	Gain
#1	6.9	8.0	8.0	8.6	8.4	12.3	8.7	8.6	8.1	8.1	8.7	8.0
#2	2.1	1.1	1.1	1.1	1.1	2.0	1.4	1.9	1.2	1.2	1.3	1.3
#3	0.8	0.8	0.8	0.8	0.8	0.1	0.8	0.8	0.8	0.8	0.8	0.8
#4	0.3	0.3	0.3	0.3	0.3	0.8	0.3	0.3	0.3	0.3	0.4	0.3
#5	0.3	0.2	0.2	0.2	0.2	2.1	0.2	0.0	0.2	0.2	0.2	0.2
#6	3.6	2.7	2.7	2.8	2.7	3.4	3.7	2.7	3.1	3.7	3.2	3.1
#7	3.0	3.7	3.7	3.9	3.7	4.2	4.0	4.0	3.9	4.4	4.2	3.8
#8	1.7	2.4	2.4	2.4	2.4	4.4	2.6	2.2	2.8	2.6	2.8	2.4
Sum	18.8	19.3	19.3	20.0	19.6	29.3	21.5	20.5	20.2	21.3	21.7	19.9

Action: Delete feature (#1)

Table 3.12: Fifth iteration of SBS

Case	Average Error Rates for Feature Deleted(%)										
	2	3	4	5	8	9	10	12	13	14	Gain
#1	6.9	6.9	7.2	7.2	7.9	7.4	7.3	6.7	7.0	7.2	6.8
#2	2.1	2.1	2.1	2.1	2.8	2.2	2.1	2.2	2.2	2.3	2.1
#3	0.8	0.8	0.8	0.8	0.1	0.8	0.8	0.8	0.8	0.8	0.8
#4	0.3	0.3	0.3	0.3	0.9	0.5	0.3	0.5	0.3	0.3	0.3
#5	0.3	0.3	0.3	0.3	1.1	0.6	0.1	0.5	0.5	0.3	0.3
#6	3.6	3.6	3.6	3.6	2.5	4.1	2.4	4.1	4.5	3.9	3.7
#7	3.0	3.0	3.1	3.0	2.8	3.2	3.0	3.2	3.4	3.4	3.2
#8	1.7	1.7	1.8	1.7	3.3	1.6	1.2	1.7	2.0	2.0	1.7
Sum	18.8	18.8	19.2	19.0	21.4	20.3	17.1	19.7	20.7	20.3	18.9

Action: Delete feature (#10)

Table 3.13: Sixth iteration of SBS

Case	Average Error Rates for Feature Deleted(%)									
	2	3	4	5	8	9	12	13	14	Gain
#1	7.3	7.3	7.8	7.2	7.2	7.6	7.2	7.5	7.6	7.4
#2	2.1	2.1	2.1	2.1	2.2	2.2	2.1	2.2	2.1	2.1
#3	0.8	0.8	0.8	0.8	0.1	0.8	0.8	0.8	0.8	0.8
#4	0.3	0.3	0.3	0.3	0.9	0.3	0.3	0.2	0.3	0.3
#5	0.1	0.1	0.1	0.1	0.4	0.3	0.2	0.2	0.1	0.2
#6	2.5	2.5	2.5	2.5	1.1	2.8	2.6	2.3	2.5	2.4
#7	2.8	2.8	3.0	2.8	1.9	2.8	2.8	3.2	2.9	3.1
#8	1.2	1.2	1.2	1.2	2.4	1.3	1.0	1.3	1.3	1.3
Sum	17.0	17.0	17.7	16.9	16.2	18.1	17.0	17.5	17.5	17.5

Action: Delete feature (#8)

Table 3.14: Seventh iteration of SBS

Case	Average Error Rates for Feature Deleted(%)								
	2	3	4	5	9	12	13	14	Gain
#1	7.2	7.2	6.8	7.0	7.3	6.2	6.4	6.8	7.6
#2	2.2	2.2	2.2	2.2	2.2	2.2	2.2	2.3	2.2
#3	0.1	0.1	0.1	0.1	0.1	0.7	0.1	0.1	0.1
#4	0.6	0.6	0.7	0.6	0.8	0.9	0.7	0.7	0.8
#5	0.3	0.3	0.5	0.3	0.7	0.5	0.5	0.3	0.5
#6	1.1	1.1	1.1	1.1	0.8	1.1	1.9	0.9	1.1
#7	1.9	1.9	1.8	1.7	2.0	2.2	2.8	2.0	2.1
#8	2.2	2.2	3.3	2.2	2.8	2.4	2.5	2.9	2.3
Sum	15.6	15.6	16.6	15.3	16.7	16.2	17.1	16.0	16.7

Action: Delete feature (#5)

Table 3.15: Eighth iteration of SBS

Case	Average Error Rates for Feature Deleted(%)							
	2	3	4	9	12	13	14	Gain
#1	7.1	7.1	7.3	7.4	6.3	6.3	6.5	7.2
#2	2.2	2.2	2.2	2.2	2.2	2.2	2.3	2.2
#3	0.1	0.1	0.1	0.1	0.7	0.1	0.1	0.1
#4	0.5	0.5	0.6	0.7	0.8	0.6	0.6	0.6
#5	0.3	0.3	0.4	0.6	0.4	0.4	0.3	0.4
#6	1.1	1.1	1.1	0.6	1.1	2.0	0.9	1.1
#7	1.6	1.6	1.6	1.6	2.1	2.8	2.0	2.1
#8	1.9	1.9	3.0	2.7	2.3	2.3	2.8	2.2
Sum	14.8	14.8	16.4	16.0	15.9	16.8	15.4	16.0

Action: Delete feature (#2)

Table 3.16: Ninth iteration of SBS

Case	Average Error Rates for Feature Deleted(%)						
	3	4	9	12	13	14	Gain
#1	6.6	7.5	7.8	5.9	6.2	6.4	7.4
#2	2.2	2.2	2.2	2.2	2.2	2.2	2.2
#3	0.1	0.1	0.1	0.7	0.1	0.1	0.1
#4	0.5	0.6	0.6	0.8	0.6	0.6	0.5
#5	0.2	0.1	0.4	0.3	0.2	0.3	0.3
#6	1.1	1.1	0.7	1.2	2.0	0.9	1.1
#7	0.9	1.6	1.3	2.1	2.8	1.7	1.9
#8	1.6	2.4	2.5	2.0	2.2	2.6	1.9
Sum	13.2	15.5	15.6	15.1	16.2	14.8	15.5

Action: Delete feature (#3)

Table 3.17: Tenth iteration of SBS

Case	Average Error Rates for Feature Deleted(%)					
	4	9	12	13	14	Gain
#1	5.5	7.2	5.6	5.4	5.7	6.5
#2	2.2	2.3	2.2	2.2	2.2	2.2
#3	0.6	0.1	0.7	0.1	0.1	0.1
#4	0.6	0.8	0.7	0.5	0.4	0.5
#5	0.4	0.4	0.2	0.0	0.1	0.2
#6	1.7	0.5	1.1	1.8	1.0	1.1
#7	1.4	0.8	1.8	2.2	0.9	1.4
#8	1.5	3.0	1.7	2.1	2.3	1.6
Sum	13.8	15.2	14.0	14.2	12.7	13.6

Action: Delete feature (#14)

Table 3.18: Eleventh iteration of SBS

Case	Average Error Rates for Feature Deleted(%)				
	4	9	12	13	Gain
#1	5.6	8.4	6.2	4.8	5.9
#2	2.2	2.2	2.2	2.1	2.4
#3	0.7	0.1	0.7	0.1	0.1
#4	0.4	0.4	0.8	0.3	0.6
#5	0.3	0.3	0.2	0.1	0.2
#6	1.4	0.6	1.1	1.7	1.1
#7	2.1	0.7	1.6	2.4	1.2
#8	1.9	2.8	2.5	2.5	2.5
Sum	14.5	15.6	15.3	13.9	13.9

Action: Stop the SBS.

Table 3.19: First iteration of SFS

Case	Average Error Rates for Feature Added(%)									
	1	2	3	5	6	7	8	10	11	14
#1	12.8	6.3	6.4	6.6	9.2	6.9	7.7	7.8	11.6	6.6
#2	1.9	2.2	2.2	2.2	4.7	3.3	2.1	2.4	5.3	2.2
#3	0.1	0.1	0.1	0.1	0.1	0.9	0.8	0.1	1.0	0.1
#4	1.0	0.6	0.6	0.6	1.3	2.6	0.2	0.5	5.2	0.5
#5	1.3	0.3	0.3	0.3	3.4	3.7	0.0	1.6	5.8	0.2
#6	3.2	0.9	0.9	0.9	6.6	5.0	2.6	2.2	8.4	1.1
#7	6.0	1.7	1.7	1.7	7.9	5.4	2.5	3.1	7.1	0.9
#8	5.9	2.6	2.6	2.3	7.6	4.4	1.1	3.2	5.9	1.6
Sum	32.1	14.7	14.8	14.7	40.7	32.2	16.9	20.9	50.3	13.2

Action: Stop the SFS.

3.3.4 Analysis of the Results

After the SBS/SFS process, a subset with 5 kinds of features has the lowest sum of average test errors. As these five kinds of features are F4 (mean of envelope, 100 dimensions), F9 (pulse duration, 100 dimensions), F12 (band width, 100 dimensions), F13(skewness of magnitude of spectrum, 100 dimensions) and Gain (1 dimension), the total number of dimensions for this subset is 401. Hereafter, we call this subset the **optimal B-Scan feature set**. Table 3.20 compares the average test error for the complete B-Scan feature data set and the optimal B-Scan feature data set for all 8 cases. Doing so shows that the SBS/SFS method does a good job of lowering the test error. In the later chapters of this thesis, unless noted otherwise, we will use the optimal B-Scan feature data as the input data.

Table 3.20: Average test errors for the B-Scan feature data sets and optimal B-Scan feature data sets

	Full B-Scan feature data sets (15 features)	Optimal B-Scan feature data sets (5 features)
Case #1	7.15%	5.7%
Case #2	5.17%	2.2%
Case #3	0.17%	0.1%
Case #4	2.31%	0.4%
Case #5	3.00%	0.1%
Case #6	5.80%	1.0%
Case #7	5.33%	0.9%
Case #8	3.52%	2.3%
Sum	32.45%	12.7%

3.4 Summary

In this Chapter, we first introduced the experimental setup for raw data collection. By applying averaging and gating techniques, A-Scan data sets and B-Scan data sets were obtained from the raw data. We then extracted features from the A-Scan data set and formed the A-Scan and B-Scan feature data sets. With fewer test errors, the B-Scan feature data set was chosen for feature reduction employing a combination of the SBS and SFS methods, we finally obtained an optimal B-Scan feature data set for further reference.

In the process of doing SBS/SFS, we used the SVM classifier for the binary classification. To simplify the test, we used the same parameters, C = infinity and $\sigma = 1000$, in the SVM classifiers, which were chosen by the grid search method. We found that appropriate parameters are critical for the SVM. Also, that it takes a lot of time to find these parameters using the grid search method. This will be demonstrated in the next chapter. Chapter 4 will focus on the selection of SVM parameters.

Chapter 4

Determination of SVM Parameters for Pipe Crack Classification

4.1 Introduction

In Chapter 3, SVM classifiers using the Gaussian kernel were employed to do feature reduction for the input vectors. In this process, we confirmed that the key factor for the performance of the SVM classifiers was the kernel parameter chosen, σ, and the trade-off parameter, C. Inappropriate parameter settings led to poor classification results [53]. In this chapter, we use an indicator called the KFD Ratio to determine the appropriate parameters for our SVM classifiers.

As noted in Section 2.3, in the algorithm of the binary SVM classification, the kernel function map the input vectors into the feature space, a higher-dimensional space, to solve a non-linear problem fitting a linear, rather than non-linear model into the feature space. The Gaussian kernel, $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\sigma^2})$, is one of the most commonly used kernels for SVM classifiers. (Refer to Table 2.2 for other kernel definitions.) The Gaussian kernel has several advantages over the kernels.

First, it has only one parameter, the bandwidth, σ ($\sigma > 0$), while the polynomial kernel, $k(\boldsymbol{x}, \boldsymbol{x}') = (\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + c)^d$, has 2 parameters, c and d. The complexity of an SVM classifier with a polynomial kernel is more than that of one with a Gaussian kernel because the former has more parameters. Also the Gaussian kernel has fewer numerical difficulties. One key point is $0 < \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\sigma^2}) < 1$. For the polynomial kernel, the kernel values may go to infinity, $(\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + c) > 1$, or zero, $(\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + c) < 1$, when the degree ($d$) is large. Second, the Gaussian kernel has features of the linear kernel. Keerthi *et al.* [53] show that a linear kernel with a trade-off parameter performs like the Gaussian kernel with certain parameters (C, σ). Third, the sigmoid kernel, $k(\boldsymbol{x}, \boldsymbol{x}') = \tanh(\kappa \langle \boldsymbol{x}, \boldsymbol{x}' \rangle + \vartheta)$ ($\kappa > 0$ and $\vartheta < 0$), which is quite popular for support vector machines because it originated from neural networks, behaves like the Gaussian kernel for certain parameters [54]. Lin *et al.* [54] recommend using the Gaussian kernel instead of the sigmoid kernel because it is hard to select suitable parameters for the sigmoid kernel. In this thesis, we have chosen the Gaussian kernel as the kernel function for our SVM classifiers.

We also discussed the trade-off parameter, denoted by C, for the misclassification error penalty in Section 2.3. C controls the trade-off between margin maximization and error minimization [55]. In total, there are two parameters, C and σ, for a binary SVM classifier with a Gaussian kernel.

Researchers have proposed several methods of finding these parameters. So far, the most conventional method is the **grid search** [56][57]. Usually, trying to exponentially grow sequences of C and σ is a practical way of identifying good parameters [56]. Besides, since there are only two parameters, the computational time needed to find good parameters using a grid search is not much greater than that using the advanced methods mentioned below. For an SVM classifier with only one or two parameters, the grid search method is a good choice. If the SVM model has more

than two parameters, the grid search must go beyond its power. Kunapuli *et al.* [58] propose a method that use bilevel programming to optimize the parameters in applications involving SVM models with many parameters. Bazi *et al.* [59] use an optimization framework based on genetic algorithms to find the parameters of SVM classifiers by using real data from the northwest Indiana's Indian Pines obtained by the AVIRIS sensor in 1992. Liu *et al.* [60] propose a hybrid method that combines evolution strategies with a grid search, to carry out an optimizing selection of these parameters using the experimental IRIS data.

In this thesis, we use an alternative indicator called the **KFD Ratio** [61] (to be defined in Section 4.2) incorporated with the grid search method to find the appropriate parameters, C and σ, for the binary SVM classifiers using the Gaussian kernel. Our aim is to reduce computational cost.

The rest of this chapter is organized as follows. In Section 4.2, the KFD Ratio is defined. We also compare the computational time for both the KFD Ratio and grid search using four benchmark data. In Section 4.3, the binary SVM classifiers which use the Gaussian kernel are applied in the experiment to different combinations of (C, σ) using the optimal B-Scan feature data sets obtained in Chapter 3. In Section 4.4, we analyze the results from the experiment and draw conclusions. Summaries are provided in Section 4.5

4.2 The Kernel Fisher Discriminant Ratio (KFD Ratio)

With different parameter σ values in the kernel function, observations are mapped into the feature space differently. For certain σ values, the data in the feature space are easier to separate. Liu *et al.* [61] use the KFD Ratio as an indicator to reflect

the separability between the two classes in the feature space. The KFD Ratio is the ratio of the **between class variance** and the **within class variance** in the feature space; that is, it is computed on the data mapped using the kernel function (we will show this below). Suppose we have two classes $\mathcal{Z}_1$ and $\mathcal{Z}_2$; the between class variance is denoted by the Euclidean distance between the two means of the classes, whereas the within class variance is denoted by the sum of the variances of the two classes.

From Chapter 2, it's acknowledged that the SVM classifier are linear classifiers. For non-linear separable data, SVMs use kernel function to map the input vectors into the feature space and do linear classification in the feature space. For example, Figure 4.1-(A) shows a classical example of a simple problem that can not be solved well using linear functions. Figure 4.1-(B) shows that after applying the mapping ϕ, the data in the feature space can be linearly separated.

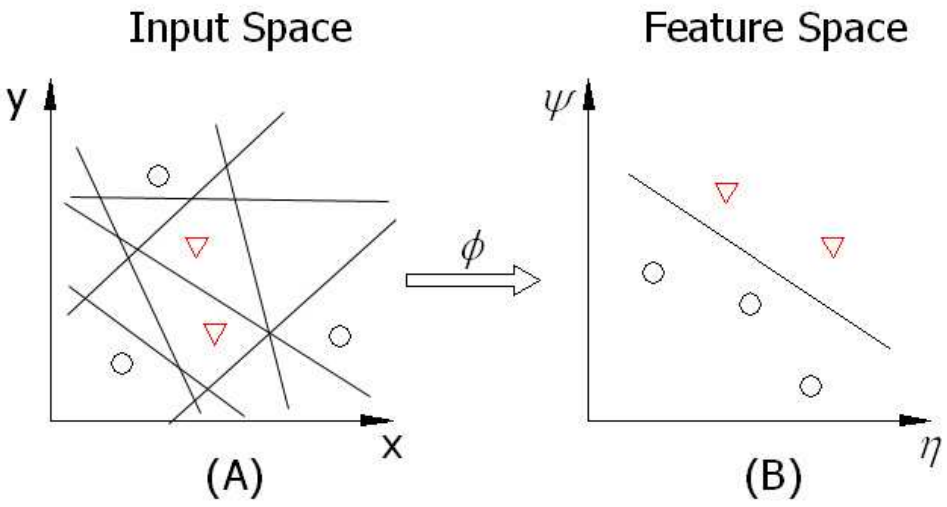


Figure 4.1: Mapping the training data nonlinearly into a feature space via ϕ

As outlined in Section 2.3, the kernel trick amounts to performing the mapping algorithm. For each kernel there exists a mapping, $\phi : \mathcal{X} \mapsto \mathcal{H}$, where $\mathcal{H}$ is the

feature space. We have $k(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}),\ \phi(\boldsymbol{z}) \rangle = \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{z})$. We are looking for a decision function of the form

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \cdot \phi(\boldsymbol{x}) + b$$

where $\boldsymbol{w} \in \mathcal{H}$ is a vector in the feature space, $\mathcal{H}$. Since the best way to work in the feature space, $\mathcal{H}$, is by using the kernel function, we endeavor to find a formulation of the KFD Ratio which uses the kernel function. In other words, the mapping function, ϕ, is used in an implicit way in the form of dot-products.

For the convenience of mathematical discussion, relevant definitions are given in Table 4.1.

Table 4.1: Nomenclature for the KFD Ratio

$\mathcal{X}$	—	the space of input vectors, $\mathcal{X} \subset \mathbb{R}^n$
y	—	the set of possible labels, $y = \{+1, -1\}$
$\mathcal{Z}$	—	training data of size l
		$\mathcal{Z} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_l, y_l)\} \subset \mathcal{X} \times y$
$\mathcal{Z}_1$	—	$\mathcal{Z}_1 = \{(\boldsymbol{x}, y) \in \mathcal{Z} \vert y = +1\}$, $l_1 = \vert\mathcal{Z}_1\vert$
		where $\vert\mathcal{Z}_1\vert$ denotes the number of data points in $\mathcal{Z}_1$
$\mathcal{Z}_2$	—	$\mathcal{Z}_2 = \{(\boldsymbol{x}, y) \in \mathcal{Z} \vert y = -1\}$, $l_2 = \vert\mathcal{Z}_2\vert$
		where $\vert\mathcal{Z}_2\vert$ denotes the number of data points in $\mathcal{Z}_2$
$\boldsymbol{m}_1$	—	$\boldsymbol{m}_1 = \frac{1}{l_1} \sum_{\boldsymbol{x} \in \mathcal{Z}_1} \phi(\boldsymbol{x})$, class mean in feature space $\mathcal{H}$ for $\mathcal{Z}_1$
$\boldsymbol{m}_2$	—	$\boldsymbol{m}_2 = \frac{1}{l_2} \sum_{\boldsymbol{x} \in \mathcal{Z}_2} \phi(\boldsymbol{x})$, class mean in feature space $\mathcal{H}$ for $\mathcal{Z}_2$
$\mathcal{S}_B$	—	between class variance matrix in feature space $\mathcal{H}$
		$\mathcal{S}_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T$
$\mathcal{S}_W$	—	within class variance matrix in feature space $\mathcal{H}$
		$\mathcal{S}_W = \sum_{i=1,2} \sum_{\boldsymbol{x} \in \mathcal{Z}_i} (\phi(\boldsymbol{x}) - \boldsymbol{m}_i)^2$

The KFD ratio was defined in [61] as

$$\text{KFD Ratio} = \frac{\mathcal{S}_B}{\mathcal{S}_W} = \frac{\| \boldsymbol{m}_1 - \boldsymbol{m}_2 \|^2}{\mathcal{S}_1^2 + \mathcal{S}_2^2} \tag{4.1}$$

where $\mathcal{S}_i$ is denoted by

$$\mathcal{S}_i^2 = \sum_{\boldsymbol{x} \in \mathcal{Z}_i} (\phi(\boldsymbol{x}) - \boldsymbol{m}_i)^2, \qquad i = 1, 2.$$

The between class variance was defined as [61]

$$\begin{aligned} \| \boldsymbol{m}_1 - \boldsymbol{m}_2 \|^2 = & \frac{1}{l_1^2} \sum_{\boldsymbol{x}_i \in \mathcal{Z}_1} \sum_{\boldsymbol{x}_j \in \mathcal{Z}_1} k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \frac{1}{2 l_1 l_2} \sum_{\boldsymbol{x}_i \in \mathcal{Z}_1} \sum_{\boldsymbol{x}_j \in \mathcal{Z}_2} k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ & + \frac{1}{l_2^2} \sum_{\boldsymbol{x}_i \in \mathcal{Z}_2} \sum_{\boldsymbol{x}_j \in \mathcal{Z}_2} k(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned}$$

and the variance of each class is

$$\begin{aligned} \mathcal{S}_i^2 &= \sum_{\boldsymbol{x} \in \mathcal{Z}_i} (\phi(\boldsymbol{x}) - \boldsymbol{m}_i)^2 \\ &= \sum_{\boldsymbol{x} \in \mathcal{Z}_i} k(\boldsymbol{x}, \boldsymbol{x}) - \frac{1}{l_i} \sum_{\boldsymbol{x}_j \in \mathcal{Z}_i} \sum_{\boldsymbol{x}_k \in \mathcal{Z}_i} k(\boldsymbol{x}_j, \boldsymbol{x}_k), \qquad i = 1, 2. \end{aligned}$$

The KFD Ratio, Eq (4.1),can be represented as

$$\text{KFD Ratio} = \frac{\sum_{i=1}^{2} \{ \frac{1}{l_i^2} \sum_{\boldsymbol{x}_j \in \mathcal{Z}_i} \sum_{\boldsymbol{x}_k \in \mathcal{Z}_i} k(\boldsymbol{x}_j, \boldsymbol{x}_k) \} - \frac{1}{2 l_1 l_2} \sum_{\boldsymbol{x}_i \in \mathcal{Z}_1} \sum_{\boldsymbol{x}_j \in \mathcal{Z}_2} k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sum_{i=1}^{2} \sum_{\boldsymbol{x} \in \mathcal{Z}_i} k(\boldsymbol{x}, \boldsymbol{x}) - \sum_{i=1}^{2} \{ \frac{1}{l_i} \sum_{\boldsymbol{x}_j \in \mathcal{Z}_i} \sum_{\boldsymbol{x}_k \in \mathcal{Z}_i} k(\boldsymbol{x}_j, \boldsymbol{x}_k) \}}. \tag{4.2}$$

In Eq (4.1), the numerator is the between class variance and the denominator is the within class variance. The larger the KFD Ratio is, the more separable the two classes are. With the introducing of the kernel function, we were able to calculate the KFD Ratio of the data in the feature space as shown in Eq (4.2). In SVM classifiers, kernel functions are used to map the data in the input space, $\mathcal{X}$, into a potential higher-dimensional feature space, $\mathcal{H}$. When both the SVM classifiers and the KFD Ratio use the same type of kernel function and take the same kernel parameters,

we suspect that the generalization performance in the SVM may be reflected by the KFD Ratio.

Usually, the procedure of training the SVM classifiers includes finding the kernel matrix and using quadratic programming [15] to do the optimization. The kernel matrix is defined as

$$K = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_l) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_l) \\ \vdots & & \ddots & \vdots \\ k(\boldsymbol{x}_l, \boldsymbol{x}_1) & k(\boldsymbol{x}_l, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_l, \boldsymbol{x}_l) \end{pmatrix} \tag{4.3}$$

where we have $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and l is the size of the training data, $\mathcal{Z}$. When we compute the KFD Ratio, we need only the kernel matrix and some very simple additions and subtractions, as illustrated in Eq (4.2). Thus if the KFD Ratio could be an indicator for choosing the SVM parameters, a lot of computational time will be saved because quadratic programming consumes much more time than the computation of the kernel matrix. We use 4 benchmark data from the UCI[1] to calculate the computational time for the SVM and the KFD Ratio. Table 4.2 shows the CPU time[2] for the SVM, which employ the leave-one-out cross validation [62] and the KFD Ratio. In Section 3.2, we talked about the cross validation. The leave-one-out is a special case which we use one sample as the test data and others as training data each time. From Table 4.2, the time for the KFD Ratio is much smaller than that for the SVM. This shows that if the KFD Ratio is an good indicator for selecting the parameters for the SVM, we will save a lot of time finding the optimal SVM classifier. In Section

[1]The UCI (University of California, Irvine) Machine Learning Repository is a collection of databases, domain theories, and data generators that are used by the machine learning community for the empirical analysis of machine learning algorithms.

[2]The configuration of the computer is: CPU: 2×AMD 2.4GHz, Memory: 2GB , Operating System: Windows 2003

Table 4.2: CPU Time for SVM & the KFD Ratio

Data Name	Training Data Sets #	Input Space Dimension #	Time for SVM t_1 (sec)	Time for KFD Ratio t_2 (sec)	t_2/t_1 (%)
Wine[a]	130	13	5.9282	0.0729	1.2
Statlog[b]	417	18	116.6331	0.0648	0.056
Spectf[c]	80	44	8.6027	0.0761	0.88
Arcene[d]	100	10000	1164.0492	14.2608	1.2

[a]Wine data uses chemical analysis to determine the origin of wines. Classes 1 and 2 are employed.

[b]Statlog data classify a given silhouette as one of four types of vehicle.

[c]Data on cardiac Single Proton Emission Computed Tomography (SPECT) images. Each patient is put into one of two categories: normal or abnormal.

[d]Arcene's task is to distinguish cancer versus normal patterns from mass-spectrometric data

4.3, we use the optimal B-Scan feature data sets from Section 3.3 as the input to show that the KFD Ratio is a good indicator for choosing parameters.

4.3 Using the KFD Ratio to Find SVM Parameters for Pipe Crack Classification

One of the goals of this thesis is to use an alternative indicator, the KFD Ratio, for identifying the parameters in SVM classifiers using the Gaussian kernel. To find the relationship of the KFD Ratio to the SVM parameters, we used the optimal B-Scan feature data sets for pipe cracks (see Chapter 3) as the input data.

The optimal B-Scan feature data sets include 8 cases (Cases $1 \sim 8$, defined in Table 3.4). Each case has 20 groups of random separations for training data and test data, defined in Table 3.6 in Section 3.3. Trying to exponentially grow sequences of C and σ is a practical way of identifying good parameters based on the grid

search method suggested in [56]. Furthermore, a coarse grid can be used first. After identifying a "better" region on the grid, a finer grid search can be conducted on that region. For all the cases, we set the trade-off parameter, C, in the SVM at 0.1, 1, 200, 5000 and infinity, which is roughly exponential for the base of 10. In each C value, we do the test for different σ values (σ = 0.1, 0.5, 1, 5, 10, 20, 50, 80, 100, 500, 1000, 10000), which is the Gaussian kernel bandwidth used as the SVM kernel function. Since we have already done a lot of tests using the optimal B-Scan data sets, we know the good parameter for σ lies in the interval $(1, 1000)$. (Refer to Chapter 3.) Thus more values in this interval are chosen for σ for the finer grid search.

For each combination of (C, σ), the average test error for each group of 20 separations is listed in Tables 4.3 to 4.7, each of which corresponds to a different value for C. For example, Table 4.3 shows the average test error for all 8 cases when C takes the value of 0.1. In this table, we let σ = 0.1, 0.5, 1, 5, 10, 20, 50, 80, 100, 500, 1000 and 10000. The average KFD ratios (Eq (4.2)) are computed for different cases (Cases $1 \sim 8$) and different Gaussian kernel bandwidths. Table 4.8 shows the result.

In this section, we calculated the average test error for different SVM parameter combinations (C, σ) for all eight cases. The results are recorded in Tables $4.3 \sim 4.7$. In the following section, we analyze these experimental results.

Table 4.3: Average test errors for different cases and kernel bandwidths at $C = 0.1$

	σ (Gaussian kernel bandwidth)											
Case	0.1	0.5	1	5	10	20	50	80	100	500	1000	10000
#1	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121
#2	0.224	0.224	0.224	0.224	0.224	0.224	0.224	0.224	0.224	0.224	0.224	0.224
#3	0.328	0.328	0.328	0.232	0.148	0.199	0.328	0.328	0.328	0.328	0.328	0.328
#4	0.448	0.448	0.448	0.165	0.098	0.259	0.448	0.448	0.448	0.448	0.448	0.448
#5	0.448	0.448	0.448	0.262	0.125	0.276	0.448	0.448	0.448	0.448	0.448	0.448
#6	0.333	0.333	0.333	0.333	0.332	0.333	0.333	0.333	0.333	0.333	0.333	0.333
#7	0.224	0.224	0.224	0.224	0.224	0.224	0.224	0.224	0.224	0.224	0.224	0.224
#8	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121

Table 4.4: Average test errors for different cases and kernel bandwidths at $C = 1$

	σ (Gaussian kernel bandwidth)											
Case	0.1	0.5	1	5	10	20	50	80	100	500	1000	10000
#1	0.121	0.121	0.121	0.123	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121
#2	0.224	0.224	0.223	0.071	0.073	0.132	0.191	0.223	0.224	0.224	0.224	0.224
#3	0.328	0.328	0.309	0.008	0.011	0.021	0.148	0.187	0.208	0.328	0.328	0.328
#4	0.448	0.448	0.367	0.047	0.053	0.055	0.106	0.263	0.269	0.448	0.448	0.448
#5	0.448	0.448	0.359	0.068	0.082	0.128	0.134	0.166	0.364	0.448	0.448	0.448
#6	0.333	0.333	0.333	0.129	0.145	0.176	0.318	0.334	0.333	0.333	0.333	0.333
#7	0.224	0.224	0.224	0.233	0.234	0.228	0.224	0.224	0.224	0.224	0.224	0.224
#8	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121	0.121

Table 4.5: Average test errors for different cases and kernel bandwidths at $C = 200$

	σ (Gaussian kernel bandwidth)											
Case	0.1	0.5	1	5	10	20	50	80	100	500	1000	10000
#1	0.121	0.121	0.122	0.110	0.100	0.096	0.138	0.126	0.120	0.121	0.121	0.121
#2	0.224	0.224	0.194	0.033	0.022	0.022	0.022	0.025	0.030	0.173	0.219	0.224
#3	0.328	0.328	0.294	0.008	0.006	0.001	0.001	0.008	0.008	0.027	0.179	0.328
#4	0.448	0.448	0.328	0.024	0.007	0.003	0.003	0.028	0.036	0.053	0.250	0.448
#5	0.448	0.448	0.316	0.035	0.020	0.003	0.010	0.025	0.043	0.134	0.146	0.448
#6	0.333	0.333	0.311	0.051	0.018	0.008	0.024	0.064	0.082	0.236	0.336	0.333
#7	0.224	0.224	0.224	0.091	0.039	0.012	0.047	0.093	0.117	0.225	0.224	0.224
#8	0.121	0.121	0.121	0.065	0.038	0.026	0.063	0.116	0.120	0.121	0.121	0.121

Table 4.6: Average test errors for different cases and kernel bandwidths at $C = 5000$

	σ (Gaussian kernel bandwidth)											
Case	0.1	0.5	1	5	10	20	50	80	100	500	1000	10000
#1	0.121	0.121	0.122	0.110	0.100	0.081	0.066	0.078	0.103	0.121	0.121	0.121
#2	0.224	0.224	0.194	0.033	0.022	0.022	0.024	0.023	0.024	0.030	0.052	0.224
#3	0.328	0.328	0.294	0.008	0.006	0.001	0.001	0.001	0.001	0.008	0.014	0.324
#4	0.448	0.448	0.328	0.024	0.007	0.003	0.003	0.003	0.003	0.040	0.051	0.328
#5	0.448	0.448	0.316	0.035	0.020	0.003	0.003	0.002	0.002	0.043	0.084	0.437
#6	0.333	0.333	0.311	0.051	0.018	0.008	0.009	0.009	0.009	0.083	0.142	0.333
#7	0.224	0.224	0.224	0.091	0.039	0.012	0.009	0.009	0.011	0.116	0.221	0.224
#8	0.121	0.121	0.121	0.065	0.038	0.027	0.022	0.022	0.022	0.120	0.121	0.121

Table 4.7: Average test errors for different cases and kernel bandwidths at C = infinity

	σ (Gaussian kernel bandwidth)											
Case	0.1	0.5	1	5	10	20	50	80	100	500	1000	10000
#1	0.121	0.121	0.122	0.110	0.100	0.081	0.065	0.057	0.063	0.057	0.057	0.060
#2	0.224	0.224	0.194	0.033	0.022	0.022	0.024	0.023	0.024	0.022	0.022	0.022
#3	0.328	0.328	0.294	0.008	0.006	0.001	0.001	0.001	0.001	0.001	0.001	0.001
#4	0.448	0.448	0.328	0.024	0.007	0.003	0.003	0.003	0.003	0.004	0.004	0.004
#5	0.448	0.448	0.316	0.035	0.020	0.003	0.003	0.002	0.001	0.001	0.001	0.002
#6	0.333	0.333	0.311	0.051	0.018	0.008	0.009	0.009	0.009	0.010	0.010	0.010
#7	0.224	0.224	0.224	0.091	0.039	0.012	0.009	0.008	0.008	0.009	0.009	0.009
#8	0.121	0.121	0.121	0.065	0.038	0.027	0.022	0.022	0.022	0.023	0.023	0.024

Table 4.8: Average KFD Ratios ($\times 10^{-4}$) for different cases and kernel bandwidths

	σ (Gaussian kernel bandwidth)											
Case	0.1	0.5	1	5	10	20	50	80	100	500	1000	10000
#1	8.40	8.39	8.63	23.43	34.25	46.18	58.09	60.46	61.05	62.11	62.14	62.15
#2	4.63	4.64	5.16	24.61	38.64	50.60	62.22	64.68	65.31	66.43	66.47	66.48
#3	3.52	3.53	3.87	27.80	46.89	59.54	71.90	74.71	75.43	76.73	76.77	76.79
#4	3.23	3.24	3.52	24.07	40.46	51.21	61.33	63.69	64.30	65.40	65.44	65.45
#5	3.23	3.23	3.51	19.37	31.96	39.78	47.27	49.14	49.63	50.52	50.55	50.56
#6	3.52	3.53	3.79	14.89	22.89	28.93	35.85	37.52	37.96	38.74	38.77	38.78
#7	4.63	4.63	4.80	12.79	19.19	24.00	29.32	30.65	31.00	31.64	31.66	31.67
#8	8.40	8.40	8.38	12.82	17.54	21.57	26.42	27.58	27.88	28.43	28.45	28.45

4.4 Analysis of the Results of the Experiment on Pipe Crack Classification

According to the results recorded in Tables 4.3 $\sim$ 4.8, which we checked further, the following observations were made.

1. From Table 4.8, we can plot Figure 4.2, where each sub-figure represents one case. In each sub-figure, we plot the values of the average KFD Ratio (Y-axis) vs. the logarithmical values of σ in the base of 10 (X-axis). For all 8 cases, in which σ lies in the interval $(0, 1)$, the KFD Ratios are small and grow slowly with σ. When σ is in the interval $(1, 100)$, the KFD Ratio values grow more quickly. When σ lies in the interval $(100, +\infty)$, the KFD Ratio values are almost stable.

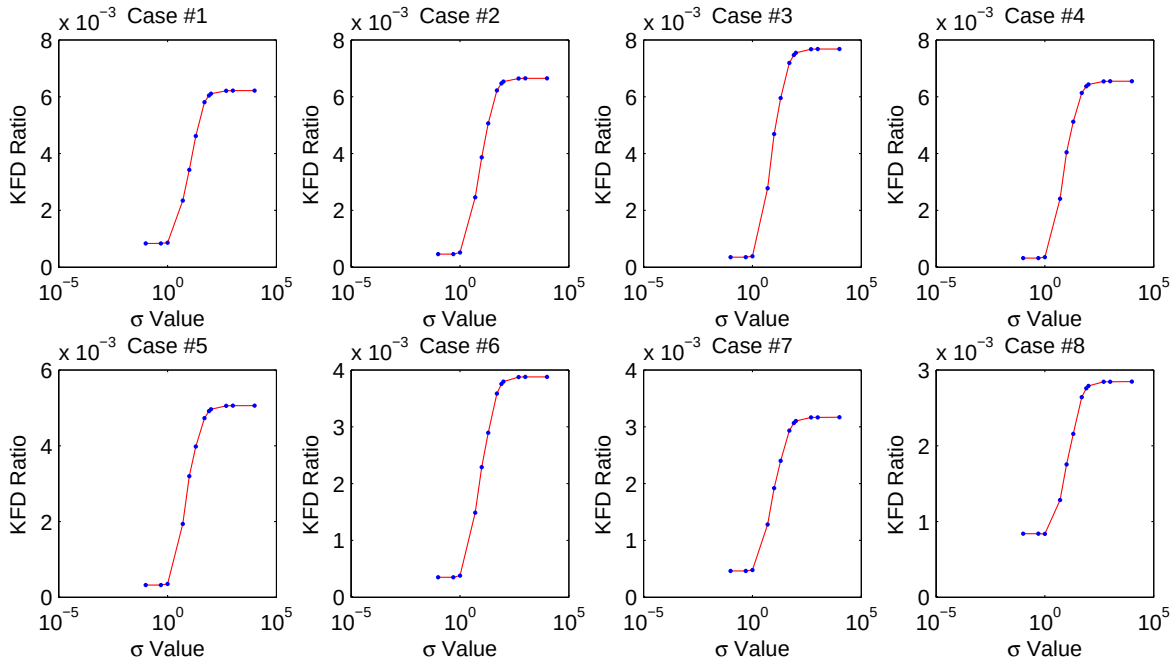


Figure 4.2: KFD Ratio values vs. σ values for different cases

2. From Tables 4.3 $\sim$ 4.7, we can plot 5 sub-figures in Figure 4.3. For example, Figure 4.3(a) corresponds to Table 4.3 at $C = 0.1$. Each sub-figure plots the value of the average test error (Y-axis) vs the logarithmical values of σ in the base of 10 (X-axis). From Figure 4.3, the minimum average test error in each sub-figure occurs in the interval $\sigma \in (1, 100)$; this matches the interval of the KFD Ratio which dramatically changes in Figure 4.2, though the errors may also be small when $C > 100$.

3. Let's look at Case #6. Figure 4.4 is based on the data from Tables 4.3 $\sim$ 4.7. Each figure plots both the value of the average test error and the value for the KFD Ratio. The minimum test error values occur in the interval $\sigma \in (1, 100)$.

Table 4.9 shows that, in each case, the minimum average test error under different C values occurs when we choose the best σ value. This table shows that when C produces different values, the minimum average test error is different. In other words, the C value affects the performance of the SVM classifiers.

Table 4.9: Minimum test errors under different C values

C value	Minimum error rate
$C = 0.1$	33.2%
$C = 1$	12.9%
$C = 200$	0.8%
$C = 5000$	0.8%
$C =$ infinity	0.8%

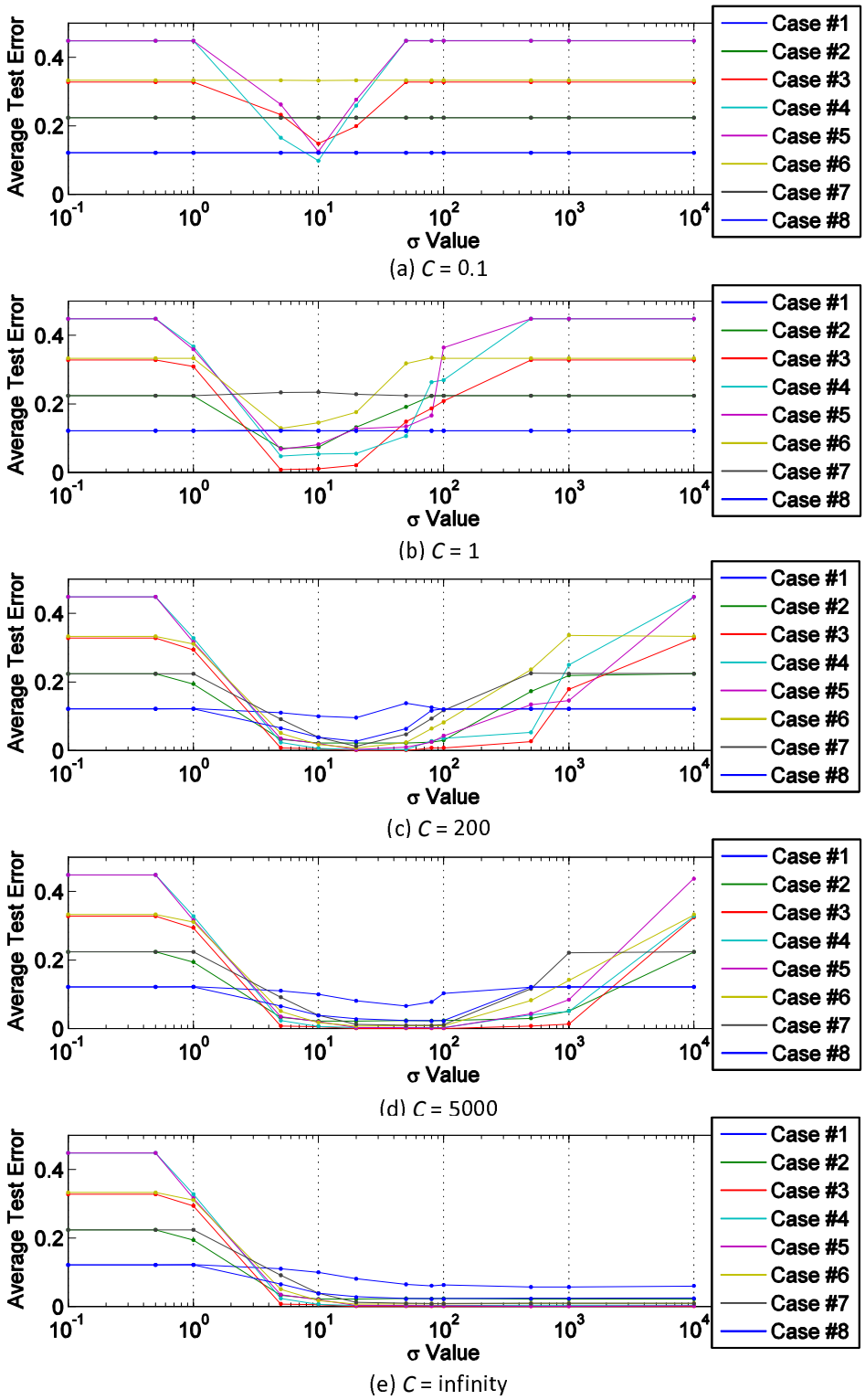


Figure 4.3: Average test errors vs. σ value for different cases

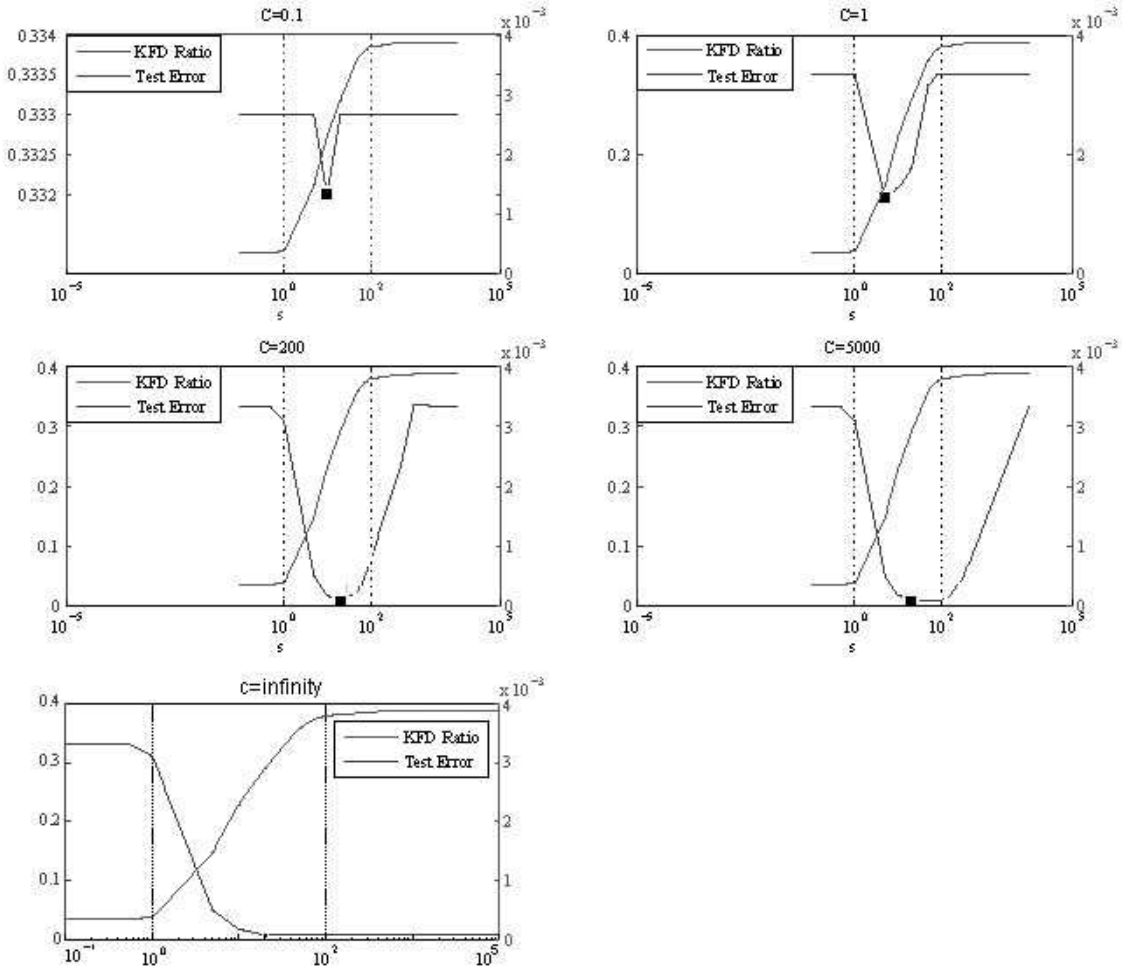


Figure 4.4: The KFD Ratio and test errors vs. σ values ($C = 0.1, 1, 200, 5000$ and infinity)

Based on the observations pointed out, we can conclude that:

1. The KFD Ratio takes much less time to calculate than the SVM classifier itself (as illustrated in Table 4.2). Whatever value is given the C parameter, the smallest classification test error occurs in the interval of σ, which increases dramatically. Before doing the grid search, we calculate the KFD Ratio. Then we can focus on the relatively small intervals, where the KFD Ratio dramatically increases, in relation to the σ value. Originally, we set σ at 0.1, 0.5, 1, 5, 10, 20, 50, 80, 100, 500, 1000, 10000. After calculating the KFD Ratio, we consider only σ =1,5,10,20,50,80,100. This can shorten the time it takes to find the parameter bandwidth, σ, of the Gaussian kernel.

2. For the optimal B-Scan feature data, different C values affect the performance, the minimum average test error, of the SVM classifiers. This thesis uses the grid search method to determine the C value.

4.5 Summary

In this chapter, we incorporated the KFD Ratio with the grid search method for choosing the parameters σ and C of the binary SVM classifiers using the Gaussian kernel. As an indicator, the KFD Ratio helps to shorten the calculating time for finding those parameters. Using the specified kernel function, the Gaussian kernel, we can find the best combination of (C, σ) for obtaining the minimum test error for the SVM classifiers. Amari *et al.* [63] proposed a data dependent method (defined in the next chapter) to improve the SVM classifiers by modifying the kernel function. If we can modify this Gaussian kernel function, we may obtain better results for the minimum test error. Based on the selected parameters of the KFD Ratio incorporated

with the grid search, in Chapter 5, we employ the data dependent method to further tune the performance of SVM classifiers.

Chapter 5

Improving SVM Performance by the Data Dependent Method

5.1 Introduction

In Chapter 4, we used the KFD Ratio incorporated with a grid search method to find the parameters of SVM classifiers using the Gaussian kernel for pipe crack data. This showed that good parameters improve the generalization performance of the SVM classifiers. Using the the KFD Ratio, we have found the best combination of (C, σ) for obtaining the smallest test error.

The general objective of classification is to find a good classifier, one which results in the smallest possible test error. In other words, the lower the test error, the better. From Tables 4.3 $\sim$ 4.7, we obtain Table 5.1, which gives the smallest average test error and the corresponding parameters (C, σ). In this table, some cases have more than one parameter set (C, σ) resulting in the smallest average test error. Compared with Cases 3$\sim$7 (refer to Table 3.4 in Section 3.2 for our definition of cases), Cases 1, 2 and 8 produce larger test errors. In the hope of obtaining a lower test error for

these three cases, a data dependent method [63] was introduced. The algorithm of the data dependent method will be briefly introduced in Section 5.2.

Table 5.1: The smallest average test error for different cases

Cases	Average Test Error	SVM Parameters (C, σ)
Case #1	5.69%	(infinity, 80)
Case #2	2.16%	(200, 10), (infinity, 10), etc.
Case #3	0.09%	(200, 20), (infinity, 100), etc.
Case #4	0.25%	(200, 50), (infinity, 50), etc.
Case #5	0.09%	(infinity, 100)
Case #6	0.80%	(5000, 20), (infinity, 20), etc.
Case #7	0.77%	(infinity, 80), (infinity, 100)
Case #8	2.16%	(5000, 50), (infinity, 80), etc.

The data dependent method is based on the selected kernel. It reconstructs this selected kernel using the so-called "conformal transformation of a kernel" technology [63] to obtain the data dependent kernel for the SVM classifiers. In Chapter 4, we found the best parameter set (C, σ) for the SVM classifiers. The data dependent method uses this Gaussian kernel as the basic kernel. By transforming this basic kernel, a data dependent kernel is provided for use as the kernel function in the

classifiers. The details of the transformation are illustrated in Section 5.2.

In this chapter, with the help of the optimized data dependent kernel, we seek to fine-tune the performance of the SVM classifiers in Chapter 4. We expect to obtain a lower test error for the classifiers. The latter part of this chapter is organized as follows. The algorithm for the data dependent method is introduced in Section 5.2. We then optimize the data dependent kernel in Section 5.3 to reduce the test error for pipe crack data. Finally, analysis and conclusions are provided in Section 5.4.

5.2 The Algorithm for the Data Dependent Method

5.2.1 The Data Dependent Kernel

Let's consider a training data set: $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_l, y_l) \in \mathbb{R}^d \times \{\pm 1\}$. We use the so-called "conformal transformation of a kernel"[63] as our data-dependent kernel function.

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = q(\boldsymbol{x}_i) q(\boldsymbol{x}_j) k_0(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{5.1}$$

where $k_0(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is called the **basic kernel**, which is an ordinary kernel such as a Gaussian or polynomial kernel. In this thesis, the basic kernel is the Gaussian kernel $k_0(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(\frac{-\|x_i - x_j\|^2}{2\sigma^2})$ (σ is the basic kernel's bandwidth) we obtained in Chapter 4. Because the KFD Ratio incorporated with a grid search enables us to find the SVM classifiers' best set of parameters (C, σ), that for which the test error is the smallest, we believe that this Gaussian kernel is a good choice. The function, $q(\cdot)$, takes the form

$$q(\boldsymbol{x}) = \alpha_0 + \sum_{i=1}^{n} \alpha_i k_1(\boldsymbol{x}, \boldsymbol{a}_i) \tag{5.2}$$

where α_i is called the combination coefficients and $k_1(\boldsymbol{x}, \boldsymbol{a}_i) = \exp(-\frac{\|\boldsymbol{x} - \boldsymbol{a}_i\|^2}{2\Sigma^2})$. $\boldsymbol{a}_i$ $(i = 1, 2, \cdots, n)$ is the **empirical core** which is selected from the training data $\boldsymbol{x}_j$

($\boldsymbol{x}_j \in \mathbb{R}^d, j = 1, 2, \cdots, l$). Since n is the number of the training data selected as the empirical core, it can be less than or equal to l. Σ is called the empirical core's bandwidth [64]. In [63], Amari *et al.* choose the support vectors as the empirical core, in order to increase the class separation. In this thesis, we also use the support vectors as the empirical core when we obtain the lowest test error using the KFD Ratio incorporated grid search method.

In the following, we deduct the matrix form for Eq. (5.1).

Using K and K_0 to denote the kernel matrices corresponding to the data dependent kernel, $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and the basic kernel, $k_0(\boldsymbol{x}_i, \boldsymbol{x}_j)$, respectively,

$$K = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_l) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_l) \\ \vdots & & \ddots & \vdots \\ k(\boldsymbol{x}_l, \boldsymbol{x}_1) & k(\boldsymbol{x}_l, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_l, \boldsymbol{x}_l) \end{pmatrix} \tag{5.3}$$

$$K_0 = \begin{pmatrix} k_0(\boldsymbol{x}_1, \boldsymbol{x}_1) & k_0(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k_0(\boldsymbol{x}_1, \boldsymbol{x}_l) \\ k_0(\boldsymbol{x}_2, \boldsymbol{x}_1) & k_0(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k_0(\boldsymbol{x}_2, \boldsymbol{x}_l) \\ \vdots & & \ddots & \vdots \\ k_0(\boldsymbol{x}_l, \boldsymbol{x}_1) & k_0(\boldsymbol{x}_l, \boldsymbol{x}_2) & \cdots & k_0(\boldsymbol{x}_l, \boldsymbol{x}_l) \end{pmatrix} \tag{5.4}$$

where K and K_0 are both $l \times l$ matrices.

Vectors $\boldsymbol{q}$ and $\boldsymbol{\alpha}$ represent $(q(\boldsymbol{x}_1), q(\boldsymbol{x}_2), \cdots, q(\boldsymbol{x}_l))^T$ and $(\alpha_0, \alpha_1, \cdots, \alpha_n)^T$, respectively. Let

$$K_1 = \begin{pmatrix} 1 & k_1(\boldsymbol{x}_1, \boldsymbol{a}_1) & \cdots & k_1(\boldsymbol{x}_1, \boldsymbol{a}_n) \\ 1 & k_1(\boldsymbol{x}_2, \boldsymbol{a}_1) & \cdots & k_1(\boldsymbol{x}_2, \boldsymbol{a}_n) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & k_1(\boldsymbol{x}_l, \boldsymbol{a}_1) & \cdots & k_1(\boldsymbol{x}_l, \boldsymbol{a}_n) \end{pmatrix}. \tag{5.5}$$

Hence, the matrix form for Eq. (5.2) is

$$\boldsymbol{q} = \begin{pmatrix} 1 & k_1(\boldsymbol{x}_1, \boldsymbol{a}_1) & \cdots & k_1(\boldsymbol{x}_1, \boldsymbol{a}_n) \\ 1 & k_1(\boldsymbol{x}_2, \boldsymbol{a}_1) & \cdots & k_1(\boldsymbol{x}_2, \boldsymbol{a}_n) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & k_1(\boldsymbol{x}_l, \boldsymbol{a}_1) & \cdots & k_1(\boldsymbol{x}_l, \boldsymbol{a}_n) \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} = K_1 \boldsymbol{\alpha}. \tag{5.6}$$

If Q is the diagonal matrix, whose diagonal elements are the $\boldsymbol{q}$ vectors,

$$Q = \begin{pmatrix} \alpha_0 + \sum_{i=1}^{n} \alpha_i k_1(\boldsymbol{x}_1, \boldsymbol{a}_i) & 0 & \cdots & 0 \\ 0 & \alpha_0 + \sum_{i=1}^{n} \alpha_i k_1(\boldsymbol{x}_2, \boldsymbol{a}_i) & \cdots & 0 \\ \vdots & & \cdots & \vdots \\ 0 & 0 & \cdots & \alpha_0 + \sum_{i=1}^{n} \alpha_i k_1(\boldsymbol{x}_l, \boldsymbol{a}_i) \end{pmatrix}$$

and we have

$$K = QK_0Q. \tag{5.7}$$

Eq. (5.7) is the matrix presentation of Eq. (5.1).

using the data dependent method, the training process consists of two steps:

1. Train the SVM using the basic kernel, K_0. This achieves the best training results, i.e., the smallest test error when applying the KFD ratio.

2. Optimize the data-dependent kernel, K, and train the SVM classifiers using this new kernel matrix, K.

We've already discussed in Chapter 4 how to train the SVM classifiers using the basic kernel, K_0. In the following section, we explain the definition of separability in the empirical feature space, which is used in Section 5.2.3 as the criterion for optimizing K to improve the performance of the classifiers.

5.2.2 Measuring Class Separability in the Empirical Feature Space

In [64], Xiong *et al.* use the following formula for measuring the class separability of the training data in the empirical feature space.

$$J = \frac{tr\ S_B}{tr\ S_W} \tag{5.8}$$

where S_B is the "between-class scatter matrix", S_W the "within-class scatter matrix", and "tr" denotes the trace of a matrix. J is also a well known Fisher scalar for measuring class linear separability, and is called criteria J_4 in [65].

Without loss of generality, let us now assume that the first l_1 data belong to class $\mathcal{Z}_1$, whose y_i is labeled as $+1$ ($i \leqslant l_1$), and the remaining l_2 data belong to $\mathcal{Z}_2(l_1 + l_2 = l)$, whose y_i is labeled as -1 ($i \leqslant l_2$). We use $\phi(x_i)$ to denote the images of the training data, X, in feature space. Also $\phi_1(x_i)$ and $\phi_2(x_i)$ are used to represent the images for the training data in $\mathcal{Z}_1$ and $\mathcal{Z}_2$, respectively. In [64], Xiong *et al.* define the S_B and S_W as follows,

$$tr\ S_B = \frac{1}{l}\sum_{i=1}^{2} l_i(\overline{\phi_i(x)} - \overline{\phi(x)})(\overline{\phi_i(x)} - \overline{\phi(x)})^T \tag{5.9}$$

$$tr\ S_W = \frac{1}{l}\sum_{i=1}^{2}\sum_{x\in\mathcal{Z}_i}(\phi(x) - \overline{\phi_i(x)})(\phi(x) - \overline{\phi_i(x)})^T \tag{5.10}$$

where $\overline{\phi(x)}$, $\overline{\phi_1(x)}$ and $\overline{\phi_2(x)}$ denote the center of the entire training data and the centers of $\mathcal{Z}_1$ and $\mathcal{Z}_2$ in the feature space.

At the same time, the kernel matrix in Eq. (5.3) can be written as

$$K = \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix} \tag{5.11}$$

where K_{11}, K_{12}, K_{21} and K_{22} represent the submatrices of K of order $l_1 \times l_1$, $l_1 \times l_2$, $l_2 \times l_1$ and $l_2 \times l_2$, respectively. In other words, Eq. (5.11) is just an ordered version

of Eq. (5.3). An ordered version of K_0 in Eq. (5.4) can be obtained in the same manner with K as follows:

$$K_0 = \begin{pmatrix} K_{11}^0 & K_{12}^0 \\ K_{21}^0 & K_{22}^0 \end{pmatrix}. \tag{5.12}$$

With the mathematical transformation (detailed proof was illustrated in [64]), Eq. (5.9) and (5.10) can be written as

$$\begin{aligned} tr\ S_B &= \tfrac{1}{l} 1_l^T \Big(\begin{pmatrix} \frac{1}{l_1} K_{11} & 0 \\ 0 & \frac{1}{l_2} K_{22} \end{pmatrix} - \tfrac{1}{l} \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix} \Big) 1_l \\ &= \tfrac{1}{l} q_l^T \Big(\begin{pmatrix} \frac{1}{l_1} K_{11}^0 & 0 \\ 0 & \frac{1}{l_2} K_{22}^0 \end{pmatrix} - \tfrac{1}{l} \begin{pmatrix} K_{11}^0 & K_{12}^0 \\ K_{21}^0 & K_{22}^0 \end{pmatrix} \Big) q_l \end{aligned} \tag{5.13}$$

$$\begin{aligned} tr\ S_W &= \tfrac{1}{l} 1_l^T \Big(\begin{pmatrix} k_{11} & 0 & \cdots & 0 \\ 0 & k_{22} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & k_{ll} \end{pmatrix} - \begin{pmatrix} \frac{1}{l_1} K_{11} & 0 \\ 0 & \frac{1}{l_2} K_{22} \end{pmatrix} \Big) 1_l \\ &= \tfrac{1}{l} q_l^T \Big(\begin{pmatrix} k_{11}^0 & 0 & \cdots & 0 \\ 0 & k_{22}^0 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & k_{ll}^0 \end{pmatrix} - \begin{pmatrix} \frac{1}{l_1} K_{11}^0 & 0 \\ 0 & \frac{1}{l_2} K_{22}^0 \end{pmatrix} \Big) q_l \end{aligned} \tag{5.14}$$

where 1_l is the $l-$dimensional vector whose entries are all equal to unity. Let B, W,

B_0 and W_0 be as follows,

$$B = \begin{pmatrix} \frac{1}{l_1}K_{11} & 0 \\ 0 & \frac{1}{l_2}K_{22} \end{pmatrix} - \frac{1}{l}\begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix} \tag{5.15}$$

$$W = \begin{pmatrix} k_{11} & 0 & \cdots & 0 \\ 0 & k_{22} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & k_{ll} \end{pmatrix} - \begin{pmatrix} \frac{1}{l_1}K_{11} & 0 \\ 0 & \frac{1}{l_2}K_{22} \end{pmatrix} \tag{5.16}$$

$$B_0 = \begin{pmatrix} \frac{1}{l_1}K^0_{11} & 0 \\ 0 & \frac{1}{l_2}K^0_{22} \end{pmatrix} - \frac{1}{l}\begin{pmatrix} K^0_{11} & K^0_{12} \\ K^0_{21} & K^0_{22} \end{pmatrix} \tag{5.17}$$

$$W_0 = \begin{pmatrix} k^0_{11} & 0 & \cdots & 0 \\ 0 & k^0_{22} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & k^0_{ll} \end{pmatrix} - \begin{pmatrix} \frac{1}{l_1}K^0_{11} & 0 \\ 0 & \frac{1}{l_2}K^0_{22} \end{pmatrix} \tag{5.18}$$

Then we can obtain the separability (Eq. (5.8)) of the training data in the empirical feature space as follows [64]:

$$J = \frac{1_l^T B 1_l}{1_l^T W 1_l} = \frac{q_l^T B_0 q_l}{q_l^T W_0 q_l}. \tag{5.19}$$

As we have already shown in Section 5.2.1 that q is a function of $\boldsymbol{\alpha}$ in Eq. (5.6), we know that J is a function of $\boldsymbol{\alpha}$ also. To maximize the separability of the two classes, we need to optimize the parameter, $\boldsymbol{\alpha}$. When we get the maximal value for J, the corresponding data dependent kernel, K, computed by Eq. (5.7), will give the best classifier performance. The next section shows how to optimize the separability, $J(\boldsymbol{\alpha})$.

5.2.3 Optimization of the Data Dependent Kernel

To maximize $J(\boldsymbol{\alpha})$, we follow the standard gradient approach. Let

$$\begin{cases} J_1(\boldsymbol{\alpha}) &= q_l^T B_0 q_l \\ J_2(\boldsymbol{\alpha}) &= q_l^T W_0 q_l \end{cases}$$

where $J_1(\boldsymbol{\alpha})$ is the numerator of Eq. (5.19) and $J_2(\boldsymbol{\alpha})$ is its denominator.

Hence we have

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = \frac{2}{J_2}(M_0 - JN_0)\boldsymbol{\alpha} \tag{5.20}$$

where $M_0 = K_1^T B_0 K_1$ and $N_0 = K_1^T W_0 K_1$. According to the general gradient method, the updating equation for maximizing the class separability, J, as given by [64] is:

$$\boldsymbol{\alpha}^{(n+1)} = \boldsymbol{\alpha}^{(n)} + \eta(\frac{1}{J_2}M_0 - \frac{J}{J_2}N_0)\boldsymbol{\alpha}^{(n)} \tag{5.21}$$

where J and J_2 are functions of $\boldsymbol{\alpha}^{(n)}$, M_0 and N_0 are two constant matrices, and η is the learning rate. To ensure the convergence of the algorithm, a gradually decreasing learning rate is adopted [64].

$$\eta(t) = \eta_0(1 - \frac{t}{N}) \tag{5.22}$$

where η_0 is the initial learning rate. N denotes a pre-specified number of iterations, and t represents the current iteration number. In our experiment, with the increasing of the iteration number, t, the learning rate, $\eta(t)$, decreases. The iteration number, N, itself is randomly chosen from the range $[1, 10000]$ since we don't know how large N should be before we do the experiment.

In this section, we have introduced the terminology and algorithm for the data dependent method. After determining the basic kernel, there are still several parameters for optimizing the data dependent kernel. These parameters include the empirical core, $\boldsymbol{a}_i$; the learning rate, η; the empirical core's bandwidth, Σ; and the

iteration number, N. In the following section, we employ the data dependent method to construct the data dependent kernel for the SVM classifiers for pipe crack data.

5.3 Optimizing the Data Dependent Kernel for Pipe Crack Data

5.3.1 The Experimental Data Set

In Chapter 4, optimal B-Scan feature data sets were used to find the best parameter set (C, σ) by the KFD Ratio method. In this chapter, the same optimal B-Scan feature data sets are used as the input vectors. As mentioned in Section 5.1, the test error for Case 1 (5.69%), Case 2 (2.16%) and Case 8 (2.16%) are relative larger than for the other cases. We use only these three cases in the experiment involving the data dependent method. As defined in Table 3.6 in Section 3.3, each of these three cases have 20 groups of random separations for the training and test data used continuously with the experimental data in Chapter 4.

5.3.2 The Experiment Using Optimal B-Scan Feature Data Sets

In Chapter 4, we used the KFD Ratio incorporated with a grid search and found the best parameter set (C, σ); that is listed in Table 5.1, for the SVM classifiers using a Gaussian kernel function. Since more than one parameter set (C, σ) corresponds to the smallest test error for the classifiers, we can choose any one of them as the basic kernel. Here, we take $(\text{infinity}, 80)$, $(\text{infinity}, 10)$ and $(\text{infinity}, 80)$ as the basic kernel parameters for Cases 1, 2 and 8, respectively.

In this experiment, the support vectors chosen as the empirical cores are those with the smallest test error in the SVM classifier using the Gaussian kernel. The initial learning rates are all set at 0.01 because thsi has been experimentally proven to ensure the convergence of the data dependent kernel under the gradient optimization algorithm. From Eq. (5.22), $\eta(t) = \eta_0(1 - \frac{t}{N})$; with a fixed η_0, $\eta(t)$ is dependent on the iteration number, N. As we have already chosen the support vectors (selected using the KFD Ratio method in Chapter 4) we have only two variables, N and the empirical core's bandwidth, Σ. We use the following method to find the combination of the empirical core's kernel bandwidth and iteration number. Let the iteration number be a random integer between 1 and 10000, and let the empirical core's kernel bandwidth be a random real number between 0.01 and 1000. If the test error is smaller than the test error provided by the regular SVM classifiers (the optimal result in Chapter 4), we stop running the program and record all the parameters. The range selection of N and Σ will be discussed in Section 5.4. As we can see, even with a step of 0.01 for Σ and 1 for N, there are $10000 \times 100000 = 10^9$ combinations required to test each group of data. Because it would be difficult to make this many attempts, we set 20000 as the total number of combinations to be tested.

For Cases 1, 2 and 8, Tables 5.2 $\sim$ 5.4 list the test error and parameters of the data dependent SVM classifiers. In those tables, test error 1 represents the test error generated by the SVM classifiers using the Gaussian kernel, and test error 2 represents the test error generated by the data dependent SVM classifiers. At the bottom of each table, an average test error is given; this was obtained by calculating the average of each test error column, separately. The analysis and discussion of the results are given in Section 5.4.

Figure 5.1 shows four examples of the gradient optimization of the data dependent method for different cases. All the x-axes in Figure 5.1 denote the iteration number

and the y-axes denote the separability $J(\boldsymbol{\alpha})$. As the iteration number increased, the value of $J(\boldsymbol{\alpha})$ increased. At the same time, the test error went from 12.07% to 10.34% in Figure 5.1(a), 12.07% to 10.34% in Figure 5.1(b), 3.45% to 1.72% in Figure 5.1(c), and 1.72% to 0 in Figure 5.1(d).

In this section, we used a data dependent kernel to optimize the SVM classifiers for the pipe crack data. The results of this experiment are listed in Tables 5.2 $\sim$ 5.4. In the next section, we analyze these results and draw conclusions.

Table 5.2: Case #1 — Test Errors & Parameters

Group #	Test Error 1	Test Error 2	η	σ	Iteration #
#1	0.0345	0.0172	0.01	10	3000
#2	0.0690	0.0690	0.01	100	3000
#3	0.0862	0.0862	0.01	500	3000
#4	0.0690	0.0690	0.01	100	3000
#5	0.0517	0.0517	0.01	100	3000
#6	0.0690	0.0690	0.01	1000	3000
#7	0.0345	0.0345	0.01	100	20
#8	0.0862	0.0862	0.01	1000	30
#9	0.1207	0.1034	0.01	1	200
#10	0.0000	0.0000	0.01	1000	30
#11	0.0690	0.0690	0.01	1000	30
#12	0.0345	0.0345	0.01	1000	30
#13	0.0517	0.0517	0.01	1000	30
#14	0.0172	0.0172	0.01	1000	30
#15	0.0690	0.0690	0.01	1000	30
#16	0.0345	0.0172	0.01	14	200
#17	0.0345	0.0172	0.01	7	200
#18	0.1207	0.1034	0.01	50	1925
#19	0.0345	0.0345	0.01	1000	200
#20	0.0517	0.0517	0.01	1000	200
Average	5.69%	5.26%			

Table 5.3: Case #2 — Test Errors & Parameters

Group #	Test Error 1	Test Error 2	η	σ	Iteration #
#1	0.0172	0.0000	0.01	45	3000
#2	0.0345	0.0172	0.01	3	2500
#3	0.0172	0.0000	0.01	14	2000
#4	0.0172	0.0000	0.01	650	300
#5	0.0345	0.0172	0.01	25	2300
#6	0.0345	0.0345	0.01	10	200
#7	0.0345	0.0172	0.01	7	2700
#8	0.0172	0.0172	0.01	10	200
#9	0.0517	0.0345	0.01	400	600
#10	0.0172	0.0172	0.01	10	10
#11	0.0345	0.0172	0.01	45	2900
#12	0.0000	0.0000	0.01	10	200
#13	0.0172	0.0000	0.01	40	1300
#14	0.0000	0.0000	0.01	10	10
#15	0.0172	0.0172	0.01	10	10
#16	0.0345	0.0345	0.01	10	10
#17	0.0172	0.0172	0.01	10	10
#18	0.0000	0.0000	0.01	10	10
#19	0.0172	0.0000	0.01	23	2900
#20	0.0172	0.0172	0.01	10	10
Average	2.15%	1.29%			

Table 5.4: Case #8 — Test Errors & Parameters

Group #	Test Error 1	Test Error 2	η	σ	Iteration #
#1	0.0690	0.0690	0.01	80	10
#2	0.0172	0.0172	0.01	80	10
#3	0.0172	0.0172	0.01	80	10
#4	0.0172	0.0172	0.01	80	10
#5	0.0172	0.0172	0.01	80	10000
#6	0.0000	0.0000	0.01	80	10
#7	0.0345	0.0345	0.01	80	10
#8	0.0172	0.0172	0.01	80	10
#9	0.0517	0.0345	0.01	40	1300
#10	0.0172	0.0172	0.01	44	800
#11	0.0172	0.0172	0.01	44	800
#12	0.0172	0.0172	0.01	44	800
#13	0.0172	0.0000	0.01	58	1000
#14	0.0000	0.0000	0.01	80	10
#15	0.0172	0.0000	0.01	50	2700
#16	0.0172	0.0172	0.01	80	10
#17	0.0345	0.0345	0.01	80	10
#18	0.0345	0.0345	0.01	80	10
#19	0.0172	0.0172	0.01	80	10
#20	0.0000	0.0000	0.01	80	10
Average	2.15%	1.90%			

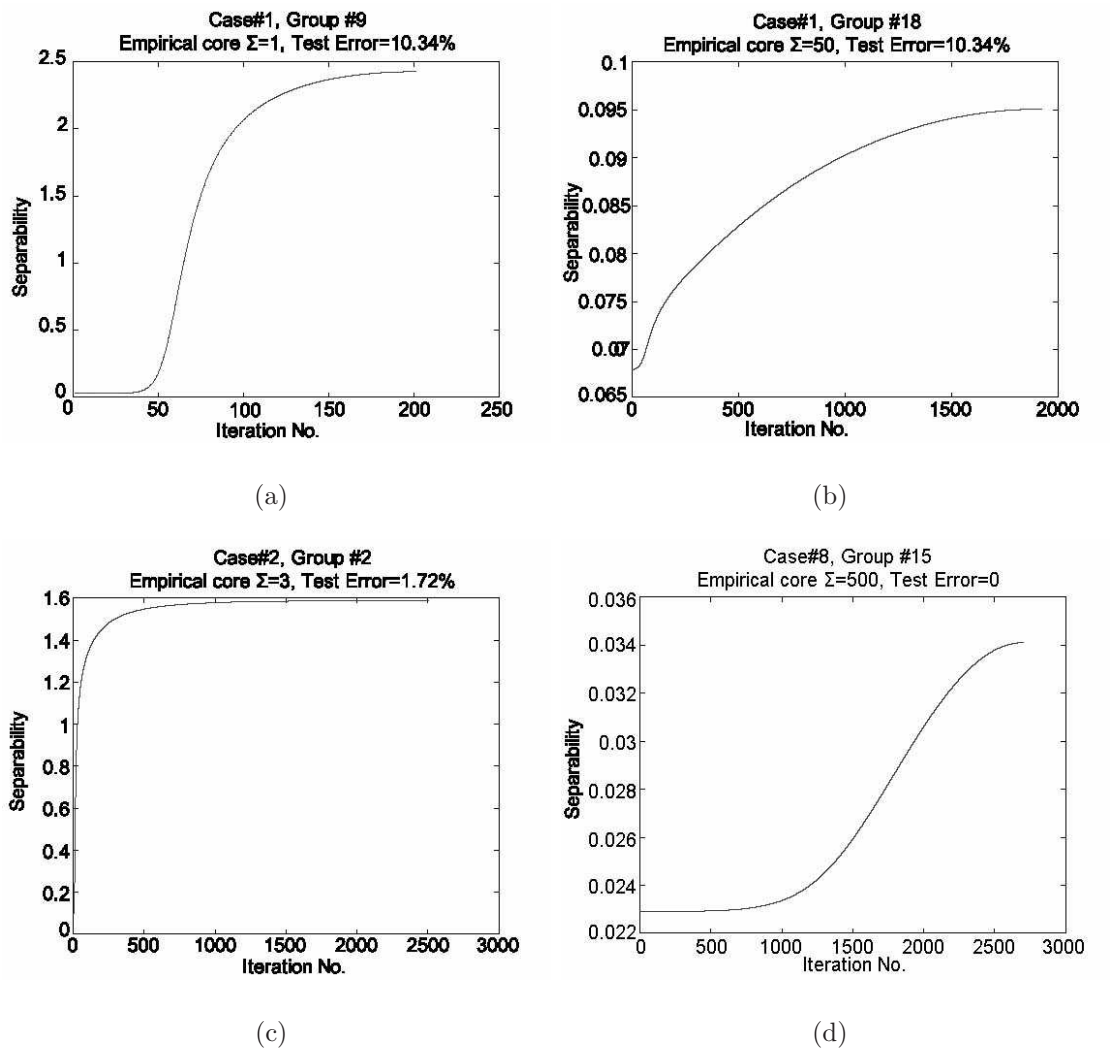


Figure 5.1: Examples of gradient optimization of separability for different cases

5.4 Analysis and Discussion of the Experimental Results

Section 5.3 listed the test errors for Case 1, 2 and 8 in Tables 5.2 $\sim$ 5.4. In this Section, we will first show our observations. Then, based on these observations, we draw the conclusions.

In Table 5.2 in Section 5.3, Test Error 2 is smaller than Test Error 1 for groups 1, 9, 16, 17 and 18, all of which are underlined. For the other groups in Table 5.2, Test Error 2 is equal to Test Error 1. The average test error for these 20 groups was lowered from 5.69% to 5.26%, or (that is by 7.56 percent $[\frac{5.69\%-5.26\%}{5.69\%}]$) by applying the data dependent method. In Table 5.3, Test Error 2 is smaller than Test Error 1 for groups 1, 2, 3, 4, 5, 7, 9, 11, 13 and 19, which are all underlined. For the other groups in Table 5.3, Test Error 2 is equal to Test Error 1. The average test error for these 20 groups was lowered from 2.15% to 1.29%; that is, b 40 percent, by applying the data dependent method. In Table 5.4, Test Error 2 is smaller than Test Error 1 for groups 9, 13 and 15, which are underlined. For the other groups in Table 5.3, Test Error 2 is equal to Test Error 1. The average test error for these 20 groups was lowered from 2.15% to 1.90%; that is, by 11.63 percent, by applying the data dependent method.

Now we can draw some conclusions based on these data

1. In three cases (Cases 1, 2 and 8), by applying the data dependent method, the average test error was reduced.

2. Since the data dependent method uses empirical cores and gradient optimization, parameters in the data dependent method such as iteration number and empirical core bandwidth vary from group to group. As mentioned earlier, the

empirical cores, $\boldsymbol{a}_i$; the learning rate, η; the empirical cores' kernel bandwidth, σ; and the iteration numbers are all new parameters in this data dependent method.

3. By applying the data dependent method, Test Error 2 should always be less than or equal to Test Error 1. The reason is that we construct the new kernel using Eq (5.1), $k(\boldsymbol{x}, \boldsymbol{z}) = q(\boldsymbol{x})q(\boldsymbol{z})k_0(\boldsymbol{x}, \boldsymbol{z})$, by the data dependent method. In Eq (5.1), $q(\boldsymbol{x})$ and $q(\boldsymbol{z})$ take the form $q(\boldsymbol{x}) = \alpha_0 + \sum_{i=1}^{n} \alpha_i k_1(\boldsymbol{x}, \boldsymbol{a}_i)$ (Eq (5.2)), where $k_1(\boldsymbol{x}, \boldsymbol{a}_i) = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{a}_i\|^2}{2\Sigma^2})$. When an empirical core's bandwidth, Σ, becomes large enough, $k_1(\boldsymbol{x}, \boldsymbol{a}_i) \to 1$. As vector $\boldsymbol{\alpha}$ is a normalized vector, $q(\boldsymbol{x}) \to 1$ and $q(\boldsymbol{z}) \to 1$. Then $k(\boldsymbol{x}, \boldsymbol{z})$ is approximately equal to $k_0(\boldsymbol{x}, \boldsymbol{z})$.

5.5 Summary

In this chapter, we used an existing tool, the data dependent kernel, to improve the performance of the SVM classifiers using the Gaussian kernel, which was selected in Chapter 4 as the basic kernel. The support vectors for the SVM classifiers using the Gaussian kernel were used as the empirical core. In this process, we randomly selected the empirical core's bandwidth and the iteration number, whenever we obtained a smaller test error. The results of the experiment show that the test error of the data dependent SVM classifiers can be reduced by the gradient optimization. In other words, we proved that building a data dependent kernel improves the SVM classifier's performance. At the same time, a practical procedure has been provided for selecting the parameters for optimizing the data dependent kernel.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we used ultrasonic pipe crack data to build SVM classifiers which separate this crack-size data into two classes: large and small. In order to correctly categorize the new data as much as possible, we first used digital signal processing techniques such as Fourier transform to extract the potential features from the raw data (see Chapter 3). Using these potential features, a combined feature reduction method, SBS/SFS (sequential backward selection and sequential forward selection), was then used for finding the best subset of those features. As a result, an optimal B-Scan feature data set was adopted as the input for the SVM classifiers for our later research work.

To lower the SVM classifiers' test error, in Chapter 4 we used the KFD Ratio (Kernel Fisher Discriminant Ratio) as an alternative indicator for choosing the SVM's kernel parameters. We did this because one of the key factors in the performance of an SVM classifier is the choice of the kernel's parameters. In Chapter 5, based on the parameters selected for the SVM classifiers, we constructed a data-dependent

kernel. This new kernel helped to improve the generalization performance of the SVM classifiers.

This thesis employed some popular data-mining methods such as SBS, SFS, the KFD Ratio and data-dependent kernel. The main contributions of this thesis can be summarized as follows.

1. KFD Ratio is an effective indicator for selecting the kernel parameters for SVM classifiers. It only takes around 1% of SVM training time to calculate KFD Ratio. This speed up SVM's training procedure.

2. In the process of the feature reduction, Combining the SBS and SFS methods to determine the optimal subset of the potential features is quick and reliable. The average test error (all eight cases) for the B-Scan feature data sets was reduced from 3.49% to 1.59% after we use the combined SBS and SFS method.

3. In constructing the data dependent kernel, we used the kernel with the best parameters as selected by the KFD Ratio. By doing many experiments, we determined how to select the empirical kernel parameter.

4. The experiments' results (See Tables 3.5, 3.20, 5.2, 5.3 and 5.4) show that the test error was reduced by use of the appropriate procedure and approaches. Using B-Scan feature data sets instead of A-Scan feature data sets makes the average test error (all eight cases) reduced from 7.15% to 3.49%; that is, by 51.19 percent. Using optimal B-Scan feature data sets instead of B-Scan feature data sets makes the average test error (all 8 cases) reduced from 3.49% to 1.59%; that is, by 54.44 percent. And using data dependent kernel makes the average test error (only Case 1, 2 and 8) reduced from 3.33% to 2.82%; that is, by 15.32 percent.

6.2 Future Work

In this work, we successfully built binary SVM classifiers for ultrasonic pipe crack data. To reduce the test error and speed up the training of the classifier, SBS/SFS was used for feature reduction; the KFD Ratio was employed as an indicator to speed up the training, and the data dependent kernel was reconstructed for improving the performance of the classifiers. To further the work done for this thesis research could be done in three areas.

1. In this thesis, the KFD Ratio was applied only to the Gaussian kernel bandwidth. We could extend this indicator by selecting kernel parameters such as the polynomial kernel.

2. We've already done the binary classification for these ultrasonic crack data. Alternatively, we could have treated this problem as a multi-class classification problem, so that, by classifying test samples according to different ranges of crack size we could develop different maintenance strategies. As SVM is a very powerful tool for solving the regression problem, we could also use it to precisely predict crack size.

3. When using the data dependent method, choosing the empirical cores ($\boldsymbol{a}_i$), the learning rate (η), the empirical cores' kernel bandwidth (Σ) and the iteration numbers is difficult. It would be helpful to find an efficient way of choosing those parameters.

Bibliography

[1] Clifford Krauss, Jeremy W. Peters, *Biggest Oil Field in US Is Forced To Stop Pumping*, New York Times, 2006

[2] Shoupeng Song and Peiwen Que, *An Effective Defect Identification Scheme in Pipeline Ultrasonic Testing*, Russian Journal of Nondestructive Testing, ISSN 1061-8309, Vol: 42, No.: 4, Pages: 255∼260, 2006

[3] OLYMPUS Corporation, Electronic references. Retrieved April 1, 2009, from *http://www.olympusndt.com/en/*

[4] Chunguo Fei, Zhengzhi Han, Qingkun Liu, *Ultrasonic Flaw Classification of Seafloor Petroleum Transporting Pipeline Based On Chaotic Genetic Algorithm and SVM*, Journal of X-Ray Science and Technology, Vol: 14, Issue: 1, Pages:1∼9, 2006

[5] Sunil K. Sinha, Fakhri Karray, *Classification of Underground Pipe Scanned Images Using Feature Extraction and Neuro-fuzzy Algorithm*, IEEE Transactions on Neural Networks, Vol: 13, NO.: 2, 2002

[6] Hossein Ravanbod, *Application of Neuro-Fuzzy Techniques in Oil Pipeline Ultrasonic Nondestructive Testing*, NDT&E International, Vol. 38, Pages 643∼653, 2005

[7] Xiaoliang Zhao, Venugopal K. Varma, Gang Mei, Bulent Ayhan, Chiman Kwan, *In-line Nondestructive Inspection of Mechanical Dents on Pipelines with Guided Shear Horizontal Wave Electromagnetic Acoustic Transducers*, Journal of Pressure Vessel Technology-Transactions of the ASME, Vol: 127, Issue: 3, Pages: 304∼309, 2005

[8] S. M. Murigendrappa, S. K. Maiti, H. R. Srirangarajan, *Experimental and Theoretical Study on Crack Detection in Pipes Filled With Fluid*, Journal of Sound and Vibration, Vol: 270, Issues: 4∼5, Pages 1013∼1032, 2004

[9] T. J. Sejnowski, M. H. Goldstein, Jr., R. E. Jenkins, B. P. Yuhas, *Combining Visual And Acoustic Speech Signals with A Neural Network Improves Intelligibility*, Advances in Neural Information Processing Systems, Vol: 2, Pages: 232∼239, 1990

[10] Patricia Melin, Diana Bravo, Oscar Castillo, *Fingerprint Recognition Using the Fuzzy Sugeno Integral for Response Integration in Modular Neural Networks*, International Journal of General Systems, Vol: 37, Issue: 4, Pages 499∼515, 2008

[11] Jalil A, Qureshi IM, Cheema TA, Naveed A, *Hand Written Character Feature Extraction Using Non-Linear Feedforward Neural Networks*,Journal of Information Science and Engineering, Vol: 21, Issue: 2, Pages: 453∼473, 2005

[12] M. Rychetsky, J. Shawe-Taylor, M. Glesner, *Direct Bayes point machines*, Proceedings of the International Conference on Machine Learning, 2000

[13] Shigeo Abe, *Support Vector Machines for Pattern Classification*, Springer, London, 2005

[14] Yoram Gat*A Bound Concerning the Generalization Ability of a Certain Class of Learning Algorithms*, Technical Report, University of California, Berkeley, No. 548, 1999

[15] V. Vapnik, 1995. *The Nature of Statistical Learning Theory*, Springer-Verlag, New York.

[16] Seong-Whan LEE, Alessandro Verri, *Support Vector Machines For Computer Vision and Pattern Recognition*, International Journal of Pattern Recognition and Artificial Intelligence, Vol: 17, No.: 3, Pages: 331∼332, 2003

[17] Bernhard Schölkopf and Alexander J. Smola, *Learning with Kernels*, The MIT Press, London, England, 2002

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

[19] Irene Kotsia and Ioannis Pitas, *Facial Expression Recognition in Image Sequences Using Geometric Deformation Features and Support Vector Machines*, IEEE Transactions on Image Processing, Vol. 16, No. 1, 2007

[20] C. A. Micchelli, *Interpolation of scattered data: distance matrices and conditionally positive defnite functions*, Constructive Approximation, 2:11-22, 1986

[21] C. Cortes and V. Vapnik, *Support Vector Networks*. Machine Learning, 20:273-297, 1995

[22] Christopher J.C. Burges, 1998, *A Tutorial on Support Vector Machines for Pattern Recognition*, Kluwer Academic Publishers, Boston.

[23] B.E. Boser, I.M. Guyon and V. Vapnik, *A training algorithm for optimal margin classifiers*, In Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, 1992. ACM

[24] M. A. Aizerman, E. M. Braverman and L. I. Rozono-er, *Theoretical foundations of the potential function method in pattern recognition learning*, Automation and Remote Control, 25:821-837, 1964

[25] Iebeling Kaastra,Milton Boyd, *Designing a Neural Network for Forecasting Financial and Economic Time Series*, Neurocomputing, Vol: 10, Issue: 3, Pages: 215∼236, 1996

[26] Matthias Kaper, Peter Meinicke, Ulf Grossekathoefer, Thomas Lingner, and Helge Ritter, *BCI Competition 2003 - Data Set IIb: Support Vector Machines for the P300 Speller Paradigm*,IEEE Transactions on Bio-medical Engineering, Vol: 51, Issue: 6, Pages: 1073∼1076, 2004

[27] Sven F. Crone, Stefan Lessmann, Robert Stahlbock, *The Impact of Preprocessing on Data Mining: An Evaluation of Classifier Sensitivity in Direct Marketing*, European Journal of Operational Research, Vol: 173, Issue: 3, Pages 781∼800, 2006

[28] Kyungmi Lee, Vladimir Estivill-Castro, *Feature Extraction and Gating Techniques for Ultrasonic Shaft Signal Classification*. Applied Soft Computing, Vol: 7, Issue: 1, Pages: 156∼165, 2007.

[29] Lester W. Schmerr Jr., *Fundamentals of Ultrasonic Nondestructive Evaluation: A Modeling Approach*, Springer, USA, 1988

[30] Peter Hauptmannn, Niels Hoppe, Alf Puttmer, *Application of Ultrasonic Sensors in the Process Industry*, Measurement Science and Technology, 2002, Vol.13: R73-R83

[31] Zhipeng Feng, Ming J. Zuo, Xiaodong wang, *Ultrasonic Signal Signatures for Pipeline Damage Identification*, University of Alberta, Technical Report, June 15, 2007

[32] Wiki Encyclopedia. Electronic references. Retrieved April 1, 2009, from http://en.wikipedia.org/wiki/Cross-validation_(statistics)

[33] R. A. Fisher. *The use of multiple measurements in taxonomic problems*. Annals of Eugenics, 7:179C88, 1936.

[34] OLYMPUS Corporation, Electronic references. Retrieved April 1, 2009, from http://www.olympusndt.com/en/ndt-tutorials/instrumententation/ascan/

[35] NDT Resource Center, Electronic references. Retrieved April 1, 2009, from http://www.ndt-ed.org/EducationResources/CommunityCollege/Ultrasonics /EquipmentTrans/DataPres.htm

[36] Yonghong Zhang, Che Chien Ng, Ajit Sahoo, Hanxin Chen, and Ming J. Zuo, *Inspection of EMD Slots on Steel Blocks Using OmniScan*, University of Alberta, Technical Report, June 15, 2007

[37] Chuxiong Miao, Yu Wang, Yonghong Zhang, Jian Qu, Ming J Zuo and Xiaodong Wang, *A SVM classifier combined with PCA for ultrasonic crack size classification*. CCECE conference paper, 2008

[38] Timothy J. Case, Robert C. Waag, *Flaw identification from time and frequency*

features of ultrasonic waveforms, IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control, 1996, Vol. 43, No. 4: 592-600

[39] R. Osterried, A. V. Kustov, G. J. Sofko, J. A. Koehler, D. André, *On The Spectral Asymmetry of Type-ii Coherent Echoes At Large Magnetic Aspect Angles*, Journal of Geophysical Research-Space Physics, Vol: 100, Issue: A6, Pages: 9707∼9715, 1995

[40] MathWorks Inc. Electronic references. Retrieved April 1, 2009, from http://www.matlab.com, Statistics Toolbox Help

[41] Sheau-Fang Lei, William A. Ahroon, and Roger P. Hamernik, *The Application of Frequency and Time-Domain Kurtosis to the Assessment of Hazardous Noise Exposures*, Journal of the Acoustical Society of America, Vol: 96, Issue: 3, Pages: 1435∼1444, 1994

[42] Tianlu Chen, Peiwen Que, Qi Zhang, Qingkun Liu, *Ultrasonic signal identification by empirecal mode decomposition and Hilbert transform*, Review of Scientific Instruments, 2005, Vol. 76: 085109-1-6

[43] Y.M. Wang, Y.H. Kang, X.J. Wu, *Application of STFT and HOS to analyse magnetostricively generated pulse-echo signals of a steel pipe defect*, NDT&E International, 2006, Vol. 39: 289-292

[44] Piervincenzo Rizzo, Ivan Bartoli, Alessandro Marzani, Francesco Lanza di Scalea, *Defect classification in pipes by Neural Networks using multiple guided ultrasonic wave features extracted after wavelet processing*, Transactions of the ASME, Journal of Pressure Vessel Technology, 2005, Vol. 127: 294-303

[45] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, *A Practical Guide to*

Support Vector Classification, Electronic references. Retrieved April 1, 2009, from http://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf, 2008

[46] Trevor Hastie, Robert Tibshirani, Jerome Friedman, *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*, Springer, New York, 2001

[47] Hongbin Zhang, Guangyu Sun. *Feature Selection Using Tabu Search Method*. Pattern Recorgnition, 35, 701∼711, 2002

[48] E. Backer, J.A. Shipper, *On the Max-Min Approach for Feature Ordering and Selection*, Proceedings of the Seminar on Pattern Recognition, Liege, 1977.

[49] W. Siedlecki, J. Sklansky, *On automatic feature selection*, Int. J. Pattern Recognition Artif. Intell. 2 (2) (1988) 197-220.

[50] W. Siedlecki, J. Sklansky, A note on genetic algorithm for large-scale feature selection, Pattern Recognition Lett. 10(11)(1989) 335-347.

[51] K. Z. Mao, *Orthogonal Forward Selection and Backward Elimination Algorithms for Feature Subset Selection*, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICSPART B: CYBERNETICS, VOL. 34, NO. 1, FEBRUARY 2004

[52] P. Pudil, Novovicova, J. Kittler, *Floating Search Methods in Feature Selection*, Pattern Recognition Lett, 15 (11), Pages: 1119∼1125, 1994

[53] S. Sathiya Keerthi, Chih-Jen Lin, *Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel*, Neural Computation, Vol: 15, Issue: 7, Pages: 1667∼1689, 2003.

[54] Hsuan-Tien Lin, Chih-Jen Lin, *A study on Sigmoid Kernels for SVM and the Training of Non-PSD Kernels by SMO-type Methods*. Technical report, Department of Computer Science, National Taiwan University, 2003

[55] Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, Sayan Mukherjee, *Choosing Multiple Parameters for Support Vector Machines*, Machine Learning, Vol: 46, Issue: 1∼3, Page:131∼159, 2002

[56] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, *A Practical Guide to Support Vector Classification*, Technical Report, Department of Computer Science, National Taiwan University, 2008

[57] Lipo Wang, Xiuju Fu, *Data Mining with Computational Intelligence*, Published by Birkhäuser, 2005

[58] G. Kunapuli, K. P. Bennett, Jing Hu, Jong-Shi Pang, *Classification Model Selection via Bilevel Programming*, Optimization Methods & Software, Vol: 23, Issue: 4, Pages: 475∼489, 2008

[59] Yakoub Bazi, Farid Melgani, *Toward an Optimal Svm Classification System for Hyperspectral Remote Sensing Image*, IEEE Transactions on Geoscience and Remote Sensing, Vol: 44, Issue: 11, Pages: 3374∼3385, 2006

[60] Ruiming Liu, Erqi Liu, Jie Yang, Ming Li, Fanglin Wang, *Optimizing the Hyper-Parameters for Svm by Combining Evolution Strategies with a Grid Search*, Intelligent Control and Automation, Vol: 344, Pages: 712∼721, 2006

[61] Bo Liu, Xiaowei Yang, Zhifeng Hao, *Binary tree Support Vector Machine based on Kernel Fisher Discriminant for multi-classification*, ADVANCES IN Neural NetworkS - ISNN 2006, PT 1, Volume: 3971, Pages: 997-1003, 2006

[62] Gavin C. Cawley, Nicola L. C. Talbot, *Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers*, Pattern Recognition, Volume 36, Issue 11, 2003, Pages 2585-2592

[63] S.Amari and S. Wu, *Improving Support Vector Machine Classifiers by Modifying Kernel Functions*, Neural Network, Vol. 12, No. 6, pp. 783-789, 1999.

[64] Huilin Xiong, M. N. S. Swamy, M. Omair Ahmad, *Optimizing the kernel in the empirical feature space*, IEEE TRANSACTIONS ON Neural NetworkS Volume: 16, Issue: 2, Pages: 460-474, 2005

[65] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. San Diego: Academic, 1990.