A study of margin setting in pattern recognition: algorithm, performance and application

Yi Wang

Follow this and additional works at: https://louis.uah.edu/uah-dissertations

Recommended Citation
https://louis.uah.edu/uah-dissertations/67

This Dissertation is brought to you for free and open access by the UAH Electronic Theses and Dissertations at LOUIS. It has been accepted for inclusion in Dissertations by an authorized administrator of LOUIS.
A STUDY OF MARGIN SETTING IN PATTERN RECOGNITION:
ALGORITHM, PERFORMANCE AND APPLICATION

by

YI WANG

A DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
The Shared Computer Engineering Program of
The University of Alabama in Huntsville
The University of Alabama in Birmingham
to
The School of Graduate Studies
of
The University of Alabama in Huntsville

HUNTSVILLE, ALABAMA

2015
In presenting this dissertation in partial fulfillment of the requirements for a doctoral degree from The University of Alabama in Huntsville, I agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department or the Dean of the School of Graduate Studies. It is also understood that due to recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this dissertation.

(student signature)  

4/27/2015  

(date)
Submitted by Yi Wang in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Engineering and accepted on behalf of the Faculty of the School of Graduates Studies by the dissertation committee.

We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate on the work described in this dissertation. We further certify that we have reviewed the dissertation manuscript and approved it in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Engineering.

Committee Chair

Date

Department Chair

College Dean

Graduate Dean
ABSTRACT

The School of Graduate Studies
The University of Alabama in Huntsville

Degree Doctor of Philosophy College/Depar. Engineering / Electrical and Computer Engineering

Name of Candidate Yi Wang

Title A Study of Margin Setting in Pattern Recognition: Algorithm, Performance, and Application.

Margin setting is a novel supervised learning algorithm for pattern recognition. In this research, we studied margin setting in terms of its algorithm, performance, and application. The algorithm and performance were studied from a new perspective: margin. Margin measures the associated distance from the data to the classification boundaries. Margin is an important design parameter that has a direct impact on the performances of margin setting. The performance impact of margin was comprehensively analyzed in two aspects. First, given the fact that margin setting generates a spherical decision boundary in pattern recognition, we proposed a novel approach that combines margin concept in the support vector machine to spherical classification in margin setting. The margin impact analysis in spherical classification was presented using probabilities of miss classification (MC) and over classification (OC). Experiments were carried out through the Monte Carlo method. The result showed that margin setting is a margin classifier whose performance tends to improve with an increased margin within a certain range. Besides, the multi-sphere strategy employed by the margin setting algorithm allowed it to achieve lower probabilities of MC, OC and non-classification than classifiers using a single sphere as its decision boundary. On the other hand, to explore margin impact on
the performances of margin setting algorithm, we analyzed and compared with the support vector machine. The margin impact on training performance and generalization were discussed theoretically. Next, experiments were conducted on artificial data sets and benchmark data sets to analyze margin impact as well. The experimental results show that training performance gets worse with an increased margin, and generalization tends to improve with an increased margin within a certain range. As a novel application of margin setting in pattern recognition, we propose a new learning-based switching filter for suppression of impulse noise in highly corrupted digital images. Margin setting detects the corrupted pixels by classification. A new filter scheme called the Noise-Free Two-Stage (NFTS) filter was employed to restore the image. The superior performance of margin setting indicates it is a powerful supervised learning algorithm that outperforms the support vector machine when applied to salt and pepper noise detection.
ACKNOWLEDGEMENTS

I would like to express my great appreciation to my advisor, Dr. W. David Pan, for his continuous support and encouragement in my studies and research work. Without his advice and wisdom, this work would not be completed.

I am deeply grateful to Dr. Jian Fu (Alabama A&M University). His insights and suggestions helped me shape my research skills. He has always been patient and encouraging in times of new ideas and difficulties. I sincerely appreciate his guidance in this work.

I would also like to express my sincere thanks to other committee members and faculty members for their comments and suggestions. They are Dr. B. Earl Wells, Dr. Jeffrey H. Kulick, Dr. Sivaguru S. Ravindran, Dr. Reza Adhami, Dr. Arie Nakhmani (The University of Alabama at Birmingham).

Finally, I would like to say a heartfelt thank you to my Mom and Dad in China, for always being there when I needed them most, for always believing in me and encouraging me to follow my academic pursuits, for always supporting me through all these years.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>xi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>xiii</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Statement of The Problem</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1 Analysis of Margin Setting Algorithm as A Margin-based Spherical Classification</td>
<td>3</td>
</tr>
<tr>
<td>1.1.2 Impact of Setting Margin on Margin Setting Algorithm and SVM</td>
<td>5</td>
</tr>
<tr>
<td>1.1.3 Learning-based Switching filter for Noise Removal</td>
<td>7</td>
</tr>
<tr>
<td>1.2 Contributions of This Dissertation</td>
<td>9</td>
</tr>
<tr>
<td>1.2.1 Contributions on Margin Definition</td>
<td>9</td>
</tr>
<tr>
<td>1.2.2 Contributions on Margin Setting Algorithm</td>
<td>10</td>
</tr>
<tr>
<td>1.2.3 Contributions on Analysis of Margin Setting Algorithm as A Margin-based Spherical Classification</td>
<td>10</td>
</tr>
<tr>
<td>1.2.4 Contributions on Setting Margin on Margin Setting Algorithm and SVM</td>
<td>11</td>
</tr>
<tr>
<td>1.2.5 Contributions on Learning-based Switching Median Filter for Noise Removal</td>
<td>12</td>
</tr>
<tr>
<td>1.3 Outline of the Dissertation</td>
<td>13</td>
</tr>
<tr>
<td>2. MARGIN</td>
<td>14</td>
</tr>
<tr>
<td>2.1 Margin-based Learning Theory</td>
<td>14</td>
</tr>
</tbody>
</table>
2.2 Margin Definition ........................................................................................................16
  2.2.1 Margin ......................................................................................................................17
  2.2.2 Margin of Training Set .............................................................................................19
2.3 Solution Region ............................................................................................................19
2.4 Margin Impact .............................................................................................................20
  2.4.1 Training performance ...............................................................................................20
  2.4.2 Generalization ..........................................................................................................21
2.5 Advantages and Drawbacks of Margin ........................................................................21

3. MARGIN SETTING ALGORITHM .................................................................................23
  3.1 Training Procedure .....................................................................................................23
    3.1.1 Partition Process ...................................................................................................24
    3.1.2 Evolution Process ..................................................................................................25
    3.1.3 Stop Condition .......................................................................................................26
  3.2 An Example of Margin Setting Algorithm ....................................................................26
  3.3 Algorithm ....................................................................................................................33

4. ANALYSIS OF MARGIN SETTING ALGORITHM AS A MARGIN-BASED SPHERICAL CLASSIFICATION .................................................................................................36
  4.1 Spherical Classification Analysis ...............................................................................36
    4.1.1 Performance Metric ...............................................................................................37
    4.1.2 Margin Impact ........................................................................................................38
  4.2 Experiment ...................................................................................................................39
    4.2.1 Probability Analysis & Monte Carlo Methods .........................................................42
    4.2.2 One-sphere Analysis ..............................................................................................43
4.2.3 Multi-Sphere Analysis .................................................................45
4.2.4 Non-Classification Analysis ......................................................49
4.3 Summary ......................................................................................50

5. IMPACT OF SETTING MARGIN ON MARIN SETTING ALGORITHM AND
SUPPORT VECTOR MACHINES ............................................................52
5.1 Motivation ....................................................................................52
5.2 Decision Boundary ........................................................................54
  5.2.1 Decision boundary of SVM ......................................................55
  5.2.2 Decision boundary of Margin Setting .......................................56
5.3 Margin .........................................................................................57
  5.3.1 Margin of SVM .......................................................................57
  5.3.2 Margin of Margin Setting ........................................................60
5.4 Margin Impact ...............................................................................63
  5.4.1 Margin Impact of SVM ............................................................63
  5.4.2 Margin Impact of Margin Setting ..............................................67
5.5 Experiment ...................................................................................68
  5.5.1 Methodology ..........................................................................69
  5.5.2 Toy Data Set ...........................................................................70
  5.5.3 Benchmark Data Sets ...............................................................74
5.6 Summary .....................................................................................78

6. LEARNING-BASED SWITCHING MEDIAN FILTER FOR NOISE REMOVAL 80
6.1 Learning-based Noise Detection ..................................................80
6.2 Noise-Free Two-Stage Filtering ...................................................82
6.3 Experiment .................................................................................................................. 85
  6.3.1 Noise Detection Analysis .................................................................................... 87
  6.3.2 Filter Scheme Analysis ....................................................................................... 91
  6.4 Summary .................................................................................................................. 102

7. CONCLUSION AND FUTURE WORK ....................................................................... 104
  7.1 Conclusion ................................................................................................................ 104
  7.2 Future Research ...................................................................................................... 105

APPENDIX A – MATLAB IMPLEMENTATION OF MARGIN SETTING

ALGORITHM .................................................................................................................. 107

REFERENCES ............................................................................................................... 142
LIST OF FIGURES

Figure 1.1 Hypersphere and Hyperplane Decision Boundary for Classification in 2-Dimensional Space ................................................................. 4

Figure 2.1 Margin and Solution Region ........................................................................ 19

Figure 3.1 Evolution Process and Partition Process of Margin Setting ..................... 24

Figure 3.2 Margin Setting Algorithm – Generate Random Points ......................... 27

Figure 3.3 Margin Setting Algorithm – Initial Prototypes and Figure of Merit ........ 28

Figure 3.4 Margin Setting Algorithm – Set Margin .................................................. 29

Figure 3.5 Margin Setting Algorithm – Mutation .................................................... 29

Figure 3.6 Margin Setting Algorithm – Prototypes for Each Generation ............... 32

Figure 4.1 Margin Impact on Spherical Classification ................................................ 37

Figure 4.2 Increasing Margin Generates More Hyperspheres .................................. 39

Figure 4.3 One-Sphere and Multi-Sphere Classification ............................................. 41

Figure 4.4 One-Sphere Analysis for Miss Classification and Over Classification ........ 44

Figure 4.5 One-Sphere Analysis for Overall Performance ....................................... 44

Figure 4.6 Multi-Sphere Strategy in Margin Setting ................................................... 46

Figure 4.7 Multi-Sphere Analysis for Miss Classification .......................................... 47

Figure 4.8 Multi-Sphere Analysis for Over Classification ........................................... 47

Figure 4.9 Multi-Sphere Analysis for Overall Performance ....................................... 49

Figure 4.10 Multi-Sphere Analysis for Non-Classification ........................................... 50

Figure 5.1 Decision Boundary of Margin Setting and SVM ....................................... 53
Figure 5.2 Hyperplane and Margins of SVM ................................................................. 55
Figure 5.3 Hypersphere and Margins of Margin Setting ............................................. 61
Figure 5.4 The Smallest Hypersphere Enclosing Two Classes of Training Set .......... 65
Figure 5.5 Classification Results of Toy data as a Function of Margin. ...................... 71
Figure 5.6 Training accuracy, Generalization as a Function of Margin for Toy Data Set 72
Figure 5.7 Margin Impact on Training Accuracy for Benchmark Data Sets .................. 75
Figure 5.8 Margin Impact on Test Accuracy (Generalization) for Benchmark Data Sets 77
Figure 6.1 MS-NFTS Impulse Noise Suppression ......................................................... 81
Figure 6.2 No Noise-free Pixels Scenario ..................................................................... 82
Figure 6.3 Flowchart of NFTS filter .............................................................................. 83
Figure 6.4 A 3 × 3 Filtering Window ............................................................................ 84
Figure 6.5 Test Images ................................................................................................. 86
Figure 6.6 Restored Image using MSN filter Compared with Other Filters ................. 98
Figure 6.7 Impulse Suppression Performance of MSN filter Compared with Other Filters for Color Image ......................................................................................... 100
Figure 6.8 Restored Image using MSN Filter Compared with Other Filters ............... 101
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 3.1</td>
<td>Margin Setting Algorithm- Figure of Merit for Blue Class</td>
<td>30</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Margin Impact on Spherical Classification</td>
<td>38</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>Gaussian Distributed Toy Data</td>
<td>40</td>
</tr>
<tr>
<td>Table 5.1</td>
<td>Theoretical Analysis of Margin Impact on Training and Generalization</td>
<td>68</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>Attributes of Toy Dataset</td>
<td>70</td>
</tr>
<tr>
<td>Table 5.3</td>
<td>Description of UCI Data Sets</td>
<td>74</td>
</tr>
<tr>
<td>Table 6.1</td>
<td>Methods Compared in Experiment</td>
<td>87</td>
</tr>
<tr>
<td>Table 6.2</td>
<td>Noise Detection Performance of Margin Setting Compared with Neural Network and SVM</td>
<td>88</td>
</tr>
<tr>
<td>Table 6.3</td>
<td>Impulse Noise Suppression Performance of MSN filter Compared with Other Methods</td>
<td>94</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 Statement of The Problem

Pattern classification recognizes patterns in data by assigning the data to different classes. The patterns of the data can be learned from “training” data whose pattern is known. In this case, the learning process is supervised learning. Supervised learning is a machine learning task that aims to predict the classification of the new data supervised by what we have learned from the known classified data. This task can also be described as a mathematical system, which inputs data and outputs a function that can be used for prediction of future unknown data. The ability to predict accurately is “generalization”. Therefore, a learning algorithm with satisfactory generalization capability is particularly desired.

Margin, as an important concept in pattern recognition, impacts the generalization of the classifier. Classifiers with larger margins tend to classify new test samples with fewer errors. Geometrically, a larger margin yields a scenario that allows more training samples to be more widely separated from the classification boundary. Therefore, new test samples are more likely to be classified correctly by this “wide” margin classification
boundary than a “narrow” margin classification boundary. However, how to control the “wideness” of margin inspires us to study and analyze the margin impact for different margin-based classifiers, or margin-based learning algorithms.

Margin setting is a novel margin-based learning algorithm. It was initially developed for pattern recognition applications [1]. Fu et al. [2-12] applied margin setting in areas such as artificial color and hyperspectral imagery. Artificial color is a biomimetic spectral sensing and processing method to obtain spectral discriminant. The spectral discriminant is classified using margin setting to meet user’s interest [3-4]. To improve the performance of artificial color method, neighborhood effects were studied [5]. Mathematical morphology was proposed as a post-processing method to deal with unclassified pixels. This method improved the quality of extracted artificial images for artificial color [6]. Then, artificial color was applied in iris recognition [7] and hyperspectral image analysis [8]. Later, designing sensitive curves with artificial color was presented using margin setting [9]. Artificial color is applied to make a smart color camera, and then fuzzy logic was employed [10-11]. Afterward, a hypersphere classifier of margin setting was extended to multiple foci hyperellipsoids classifier, revealing a dramatic improvement in performance [12]. Most recently, a joint spectral-spatial filter using margin setting was proposed to recognize the shape, size, pose, and location of a target in a scene [13].

Margin impacts of margin setting algorithm were analyzed both theoretically and experimentally in this thesis. We studied the margin impacts in two aspects. First, different from popular linear classifiers like the support vector machine (SVM), margin setting is a spherical classifier. Hence, margin impacts on the basic spherical
classification were discussed. This work forms the main building block for margin impacts on the performances of the complex margin setting algorithm. Second, margin impacts of margin setting were comprehensively discussed and compared with SVM. The results gave us a clear guidance about how to control margin and get optimal performance. Besides, we also proposed a novel noise removal method using margin setting.

1.1.1 Analysis of Margin Setting Algorithm as A Margin-based Spherical Classification

Margin setting produces hypersphere decision boundaries using a stochastic method. This method iteratively finds the optimal decision boundaries. Decision boundaries of margin setting can be adjusted by a very important parameter called margin. Margin in margin setting is measured as the ratio that the radius of hypersphere shrinks. For example, 0.1 margin is obtained after the radius is reduced 10%. The magnitude of margin can affect the classification accuracy in both training and testing phase. Thus, margin has an impact on generalization, and thus margin setting is also a margin-based learning algorithm. Another popular margin-based algorithm is the Support Vector Machines (SVM) [15-24]. SVM employs a linear programming or quadratic programming to find the maximal margin. Different from margin setting, SVM is a linear hyperplane classifier.

The hypersphere is one fundamental boundary form for pattern recognition. It is attractive because of its simplicity in implementation and generality in classification. Hypersphere decision boundary is implemented by comparing the radius of the hypersphere with the Euclidean distance between an unknown data sample and the center
of the hypersphere, to determine if the unknown falls into the hypersphere or not. When it comes to generality in classification, hypersphere is also good at fitting the training data and achieving good generalization. In particular, hypersphere is more suitable for linear non-separable data because of its non-linear nature. Figure 1.1 shows an example of using hypersphere and hyperplane to classify red points from green points. When comparing the plots (a) and (b) in Figure 1.1, it is seen that hypersphere yields a higher true positive rate and a lower false positive rate than hyperplane for this particular case.

![Figure 1.1](image)

**Figure 1.1** Hypersphere and Hyperplane Decision Boundary for Classification in 2-Dimensional Space (a) Hypersphere classifies red points with the true positive rate of 90%, and the false positive rate of 20%. (b) Hyperplane classifies red points with the true positive rate of 80%, and the false positive rate of 15%.

Intuitively, if the circle in Figure 1.1 (a) is moved or resized, the accuracy should be changed. Therefore, to understand how the spherical classification performance is affected by the locations and sizes of the circles, an analysis of margin-based spherical classification was conducted. There are some related work on spherical classification is
discussed. Cooper et al. pointed out that the hypersphere is optimum for large classes of distributions [25]. Wang et al. proposed to construct a single separating hypersphere to classify patterns in the feature space induced by the kernel mapping from original input space [26]. To the best of our knowledge, there is no prior work on analyzing the impact of margins on the Margin Setting algorithm. We conducted a series of experiments using the Monte Carlo method on Gaussian distributed artificial data. The results show that the good performance of the Margin Setting algorithm has to do with its use of multiple hyperspheres as decision boundaries. Besides, its classification performance improves if we increase the margin, within a certain range.

1.1.2 Impact of Setting Margin on Margin Setting Algorithm and SVM

Here we studied two examples of supervised pattern classification: margin setting [1-14] and support vector machine (SVM) [15-24]. Margin setting is a new algorithm that has not been studied in great depths. It was applied to hyperspectral image processing and artificial color discriminant analysis. SVM has been shown to be successful in a variety of science and engineering fields. These two algorithms were developed from different backgrounds, but they both share the same algorithm parameter: margin. Margin works as a parameter that influences generalization of classifiers. These classifiers are called margin-based algorithm [27-31]. Some well-known margin-based algorithms are SVM, AdaBoost [32-37] and Voted-perceptron [38-42]. The goal of SVM is to maximize the minimum margin of the training examples. On the other hand, AdaBoost is a large margin classifier that does not maximize the margin explicitly. Instead, it minimizes an exponential loss function of the margins using a greedy procedure. This procedure attempts to increase margin, especially when the initial margin is very small. Voted-
perceptron, similar to SVM, is a maximal margin algorithm. The difference is that voted-perceptron uses a gradient descent approach that iteratively increases margin to the maximum, while SVM solves an optimization problem using linear programming or quadratic programming to get maximal margin. Very roughly speaking, margin measures the confidence of the classifier’s prediction, i.e., generalization. Thus, it is crucial to note that generalization performance is affected by margin. How margin affects generalization performance inspires our work. Moreover, some recent comparison work has been done by SVM, Neural Network, AdaBoost and other classification algorithms. Romero et al. compare SVM with Neural Networks with similar hidden-layer weights [43]. Shao et al. compare SVM with Neural Networks for the land-cover classification using limited training data points [44]. Morra et al. compare AdaBoost with SVM through an application in automated hippocampal segmentation [45]. SVM was also compared with maximum likelihood classification by Mondal et al. [46]. Therefore, comparing SVM with another similar margin-based algorithm, margin setting, is appealing.

Another motivation for margin-based comparison is to determine the margin impact of margin setting, so as to guide us how to tune margin and get better performance. For this purpose, it is necessary to theoretically present a common margin definition used for comparing margin setting and SVM as margin-based algorithms. One related work by Caulfield et al. [14] only discussed a limited margin impact on generalization of margin setting. The impact analysis was not adequate due to the fact that margin impact was shown within a very small number of iterations, called classification rounds. Another drawback is that they failed to give a theoretical analysis of margin impact and its connections to training performance and generalization. To this end, we present a margin
definition for general margin-based algorithms. We not only comprehensively analyze the margin impact in theory, but also conduct extensive experiments with toy data sets and benchmark data sets.

1.1.3 Learning-based Switching filter for Noise Removal

Impulse noise is generated during image transmission when there are defects in CCD elements and flecks of dust inside the camera. Salt-and-pepper noise is one important category of impulse noise. Grayscale images contaminated by salt and pepper noise are characterized by the appearance of white and black dots. White dots are salt noise with the maximum intensity value in the dynamic range while black dots are pepper noise with the minimum intensity value in its range. The pixel corrupted with noise is called noise pixel. For an 8 bits/pixel image, the typical intensity value for pepper noise is 0; for salt noise, it is 255. The corruption of color images with salt-and-pepper impulse noise is viewed as the appearance of color dots. Color dots are caused by noise in color channels. An RGB image is composed of red, green and blue channels. Thus, pepper noise with intensity 0 and salt noise with intensity 255 are added in red, green, and blue channels randomly and independently [49].

Several methods have been introduced to remove noise in grayscale and color images [48, 49, 97]. However, they only remove noise in low noise densities (≤20%) [48]. Some methods deal with high noise densities (≤80%), but they do not remove noise from images with a noise density larger than 80% due to the possible destruction of image details [49]. In order to improve the image quality, we should detect and remove the salt and pepper noise. Ideally, noise can be removed without changing the non-noise pixels. Therefore, it is desirable to suppress the noise and preserve the integrity of image
details and edges as well [50, 98, 99]. Standard median filter (SMF) is the classical non-linear filter used to remove salt and pepper noise. However, SMF removes noise effectively with low noise density that is less than 50%. When the noise density is higher than 50%, SMF produces edge jitters and a blurring effect [51]. It also fails to preserve details and fine lines. These drawbacks come from the fact that SMF filters operate on the entire image unconditionally. Therefore they alter the non-noise pixels unintentionally. Hence, a noise detection process is necessary that identifies the noise pixels and leaves non-noise pixels unchanged. To overcome the drawbacks, “decision-based” or “switching” strategy is usually used. This method differentiates noise pixels from non-noise pixels. Decision-Based Algorithm (DBA) was developed to detect and remove the high density impulse noise [52]. Esakkirajan et al. [53] introduced a modified decision-based approach that utilizes an unsymmetrical trimmed median filter. This method has achieved a better denoising result than DBA. Recently, a method [54] was introduced that claimed a switching filter can detect up to 100% of the noise by using the local extrema in the filter window for high noise densities.

It is a challenging task for decision-based or switching methods to define a robust mechanism to detect the noise. Although some approaches can detect noise pixels with a high accuracy for highly corrupted images, over detection rates and miss detection rates are still very high for low noise density images [55]. Moreover, these methods do not preserve edges very well. To overcome these drawbacks, unsupervised and supervised learning algorithms have been applied to remove salt-and-pepper noise, such as the clustering algorithm and support vector machine (SVM) [56, 57, 100]. However, both the proposed denoising-based clustering algorithm [100] and SVM algorithm [56-57] can
only remove low-densities (≤50%) of salt-and-pepper noise. Other supervised learning algorithms perform even worse than SVM. For instance, Decision Trees and Naïve Bayes yield lower predictive accuracy than SVM. The Nearest Neighbor algorithm only has satisfactory performance in low dimensions. Discriminant analysis needs to satisfy the modeling assumptions to guarantee its accuracy. Given the limitations of the existing supervised learning algorithms, we proposed to use margin setting algorithm to detect the salt-and-pepper noise. Salt-and-pepper noise pixels and non-noise pixels are randomly selected for training. During the training procedure, margin setting generates decision surfaces to classify and detect noise and non-noise pixels. Moreover, a novel noise-free two-stage filter is used to restore the image. The novelty of combining MS learning-based detection and filtering scheme NFTS leads to the superiority of the proposed switching median filter, MSN. Compared with the other existing learning-based filters, MSN performs better with a higher Peak Signal-to-Noise Ratio (PSNR), lower mean square error (MSE), higher image enhancement factor (IEF), and higher Structural Similarity Index (SSIM).

The rest of the chapter is organized as follows. An overview of our proposed methods and contributions are given in Section 1.2. Finally, in section 1.3, we present the outline of the dissertation.

1.2 Contributions of This Dissertation

1.2.1 Contributions on Margin Definition

Margin has been defined by a lot of researchers in different margin-based algorithms. We summarize them and give a comprehensive view of margin definition,
which is necessary for our analysis of margin impact on margin setting and SVM. Details will be covered in Chapter 2, and we summarize the contributions here.

- Margin definition is discussed in margin-based learning theory.
- Margin definition is explained comprehensively.
- The concept of solution region is used for analyzing the impact of margin.
- The impact of margin is analyzed in two aspects: training performance and generalization.

1.2.2 Contributions on Margin Setting Algorithm

Margin setting algorithm, as a novel supervised learning, margin-based algorithm, should be presented in detail before carefully studying the algorithm, performance, and application. The details are shown in Chapter 2, and our contributions are summarized as follows.

- Margin setting algorithm is mathematically presented, and the training procedure is discussed in detail.
- Margin setting algorithm is explained in steps using graphs and flowcharts.

1.2.3 Contributions on Analysis of Margin Setting Algorithm as A Margin-based Spherical Classification

Spherical classification uses hypersphere as decision boundary. Margin setting is a new learning algorithm for spherical classification. A novel fundamental margin impact analysis for margin setting is proposed. We analyze margin impact using the probability of miss classification (MC) and over classification (OC). Experiments are carried out through Monte Carlo method. The details will be covered in Chapter 4. Here, we summarize our contributions as follows.
• A novel approach that combines margin concept in the support vector machine to spherical classification in margin setting is proposed to analyze margin impact.

• The performance metrics are analyzed using Monte Carlo method. The result showed that margin setting is a margin-based classifier whose performance tends to improve with an increased margin within a certain range.

• Margin impact is analyzed for one-sphere and multi-sphere cases, respectively. The multi-sphere strategy employed by the margin setting algorithm allows it to achieve lower probabilities of miss classification, over-classification and non-classification than classifiers using a single sphere as its decision boundary.

1.2.4 Contributions on Setting Margin on Margin Setting Algorithm and SVM

We comprehensively compare the margin impacts on training performance and generalization, both theoretically and experimentally. In the theoretical analysis, motive and decision boundary of margin setting and SVM are discussed and compared to explain the motives for margin-based comparison. Given both margin setting and SVM can be viewed as discriminant functions, standard margin definitions are given generally for discriminant functions and then we extended specific definitions to margin setting and SVM, respectively. Likewise, margin impacts are analyzed for discriminant function, and then impacts are discussed particularly for margin setting and SVM. The details will be covered in Chapter 5. Here, we summarize our contributions as follows.

• The impact of margin on performances of margin setting algorithm is comprehensively compared with the support vector machine.
• The comparison analyzes and compares how margin affects training performance and generalization, both theoretically and experimentally using toy data set and benchmark data sets.

• The experimental results support the theoretical analysis, showing that with an increased margin, training performance get worse, and generalization tends to improve within a certain range.

1.2.5 Contributions on Learning-based Switching Median Filter for Noise Removal

A novel switching median filter integrated with a learning-based noise detection method was proposed for suppression of impulse noise in highly corrupted color images. The noise detection method employs margin setting to detect noise pixels. Margin setting detection is achieved by classifying noise and non-noise pixels with a decision surface. The decision surface is generated after the training process in Margin setting. Margin setting detection yields very high detection accuracy, i.e., a zero miss detection rate and a fairly low over-detection rate for a wide range of noise levels varying from 5% to 95%. After a pixel is detected as a noise pixel, a new filter scheme called the noise-free two-stage (NFTS) filter is triggered to correct it. NFTS corrects the noise pixels using the median of the noise-free pixels in its localized window in two stages. The results indicate that margin setting outperform SVM when it is applied to noise removal. The details will be covered in Chapter 6. Here, we summarize our contributions as follows.

• A new learning-based switching filter using margin setting algorithm is proposed.

• A novel noise-free two-stage filter is proposed to filter the noise.
• Noise removal performance is compared with other existing approaches. The results show that margin setting algorithm outperforms SVM and Neural Networks for a wide range of noise densities.

1.3 Outline of the Dissertation

The rest of the dissertation is organized as follows:

• Chapter II comprehensively presents margin definition and impact of margin on performance using the solution region.

• Chapter III presents the margin setting algorithm using mathematical formulations and flowchart.

• Chapter IV presents a margin impact analysis for spherical classification, which lays the solid foundation for analyzing margin impact on margin setting algorithm.

• Chapter V presents analysis on margin impact of the margin setting algorithm by comparing with SVM. In particular, we compare and analyze margin impact theoretically and then use toy data sets and benchmark data sets to validate our theoretical analysis.

• Chapter VI presents a novel learning-based switching median filter using margin setting. It is a new application of margin setting for removing impulse noise. The experimental results are compared with neural network and SVM.

• Chapter VII concludes our work and discusses the directions for future work.
2.1 Margin-based Learning Theory

Margin-based learning theory is based on the function estimation problem and the risk minimization problem in statistical learning theory [57, 89].

Function estimation problems can be subdivided into three categories: generator, supervisor, and learning machine. A generator can produce several data points in the form of vectors $\mathbf{x}$, with an unchanged but unknown distribution. The supervisor produces the output vector $\mathbf{y}$ based on every input of $\mathbf{x}$, and it also conforms to an unchanged but unknown conditional distribution $P(\mathbf{y}|\mathbf{x})$. The learning machine implements a series of functions $f(\mathbf{x}, \alpha)$, where $\alpha \in \Lambda$. $\Lambda$ is a set of parameters. The parameters can be scalar quantities, vectors, or abstract elements. During the learning process, one of the functions is chosen. This chosen function corresponds to the supervisor’s response, with the best approximation based on the training set $T$. The definition is given as follows.

Given a n-dimensional feature vector $\mathbf{x}, x \in R^n$, and its binary classification label $y = \{1, -1\}$ or K-class classification $y = \{1, 2, ..., K\}$ ($K > 2$), a training set $T$ is a set of training examples that are independent identically distributed (i.i.d.). It is denoted as:
\[ T = \{(x_1, y_1), ..., (x_i, y_i), ..., (x_m, y_m)\}, 1 \leq i \leq m. \]  

Equation 2.1

where \( m \) is the number of training examples, \( x_i \) is the training example and \( y_i \) is the label. For the training examples, they satisfy \( P(x, y) = P(x)P(y|x) \).

The risk minimization problem is used to measure the minimum difference, or loss, between the response of supervisor and the learning machine. The expected value of loss is computed by the risk function. The risk minimization function is defined as:

\[ R(\alpha) = \int L(y, f(x, \alpha))dP(x, y). \]  

Equation 2.2

The difference or loss is represented as \( L(y, f(x, \alpha)) \). In this case, the response of the supervisor is \( y \), and the response of learning machine is \( f(x, \alpha) \). The risk minimization is realized by finding the function \( f(x, \alpha_0) \). In these cases, \( f(x, \alpha_0) \) outputs the minimal \( R(\alpha) \) with respect to training set \( T \) and unknown distribution \( P(x, y) \).

The pattern recognition problem is learned from the labeled training set and assigns labels to unknown data with a minimal classification error. Let us consider binary classification cases for simplicity. As for binary classification, the training set in equation (1) is labeled as two classes. In this case, the output is \( y = \{-1, 1\} \). As a result, the loss function \( L(y, f(x, \alpha)) \) is defined as follows

\[ L(y, f(x, \alpha)) = \begin{cases} 0 & \text{if } y = f(x, \alpha) \\ 1 & \text{if } y \neq f(x, \alpha). \end{cases} \]  

Equation 2.3

This loss function \( L \) calculates the classification error when the supervisor’s output \( y \neq f(x, \alpha) \). An example of a pattern recognition problem is to find one of the functions, \( f(x, \alpha_1) \), implemented by the learning machine. This function has the smallest classification error with respect to the training set \( T \).

Here we define margin and margin-based learning algorithm in learning theory.
Definition 2.1 (Margin): Margin $\rho$ is a non-negative real value function. The magnitude of $\rho$ is associated with the confidence of a prediction $f(x, \alpha)$ for a given input training set $T$. A specified learned hypothesis with respect to $T$ is selected from $H$, i.e., $h \in H$. $H$ is a hypothesis space, containing a series of functions $f(x, \alpha)$ implemented by the learning machine.

Definition 2.2 (Margin-based Learning Algorithm): Given a training set $T$, a margin-based loss function $L(\rho(y, f(x, \alpha)))$ measures the difference between the predicted output and true output based on the magnitude of margin $\rho$. A learning algorithm $\Lambda$, outputs a specified learned hypothesis $h'$, selected from a given hypothesis space $H$, $h' \in H$. Here, $h'$ yields the minimal expected loss $L$, i.e.,

$$h' = \min_{h \in H} \int L(\rho(y, f(x, \alpha)))dP(x, y)$$

Equation 2.4

2.2 Margin Definition

Margin-based learning theory forms the main building block for margin definitions for real pattern recognition problems. Margin definition has been introduced by several researchers. In essence, their definitions are the same, but slightly different by adding constraints or putting in special cases. The literature [19][68] defines margin in a particular case for binary classification where label $y = \{1, -1\}$ only. However, literature [60] generalizes the margin definition to linear discriminant functions, indicating that margin has no relations with the value of class labels. In particular, margin is defined as $b$, where $a^ty \geq b$, and $y$ is the training example in the mapped feature space, $a^t$ is the normal vector to the separating plane. Especially for multi-class labels $y = \{1, 2, ..., K\}$ ($K > 2$), the class labels can add wrong weights on the magnitude of margin. In this case, comparing margins of different training examples is not correct. Literature
[30] defines margin by setting constraint $||w||_p = 1$, $p$ can be arbitrary positive number, representing p-norm matrix norm. For example, $p = 2$ is for Euclidean norm. In this case, the defined margin is equal to the value of margin’s definition in geometry.

Margin in the margin-based algorithm can be defined as a non-negative real-valued function, and its magnitude is associated with confidence in the prediction. Besides, good confidence of prediction leads to good generalization of the margin classifier. Thus, the margin may improve the generalization ability.

2.2.1 Margin

Definition 2.3 (Function Margin): Margin with respect to a hypothesis $h_m \in \mathcal{H}$ and concept $c$ is defined as a function as

$$ M = \alpha_m h_m(x_i). $$  

Equation 2.5

Where $M$ measures how far the distance from training example $x_i$ to $h_m$, where $1 \leq i \leq N$. Suppose there are $N$ training examples. $\alpha_m$ is normal to $h_m \in \mathcal{H}$.

Hypotheses $\mathcal{H}$ are the solutions of machine learning tasks, where $\mathcal{H} = \{h_1, ..., h_m, ..., h_M\}$, $1 \leq m \leq M$. Only one base hypothesis $h_m \in \mathcal{H}$ is selected by the machine learning algorithm to approximate the target concept $c$ given training set $T$.

Concept $c$ is defined by concept learning [59], an example of machine learning. In particular, a Boolean function $C(x)$ is used for a given finite instance space $X$, where $C(x) = 1$ if and only if $x \in c$, otherwise $C(x) = 0$.

Margin of hypothesis $h_m \in \mathcal{H}$, is the minimum margin of all $N$ training samples:

$$ \min \left[ \alpha_m h_m(x_1), ..., \alpha_m h_m(x_i), ..., \alpha_m h_m(x_N) \right] $$  

Equation 2.6

Margin of the final hypothesis is the maximal margin of all hypotheses $\mathcal{H} = \{h_1, ..., h_M\}$:
\[ \text{Max min}_{n=1,...,N} \sum_{m=1}^{M} \alpha_m h_m(x_n) \]  

Equation 2.7

The idea of function margin of the final hypothesis is a max-min problem that maximizes the minimal margins of all training samples.

One example of hypotheses is linear classifiers. Margin can be defined as a function of linear classifiers. Given a weight vector \( w \), a training example \((x_i, y_i)\), an arbitrary function of \( x_i \) denoted as \( \varphi(x_i) \), margin with respect to training example \((\varphi(x_i), y_i)\) is defined as the absolute value of a linear combination of all features of \( \varphi(x_i) \):

\[ m(x_i) = |w^T \varphi(x_i) + b|, \]

\[ \varphi(x_i) = \begin{cases} x_i & \text{Linear separation} \\ \text{others} & \text{Nonlinear separation} \end{cases} \]  

Equation 2.8

Where \( x_i = \{x_{i1}, x_{i2}, ..., x_{id}\} (d \geq 1) \), a \( d \)-by-1 vector containing \( d \) features in \( V \) space. \( \varphi(x_i) = \{\varphi(x_{i1}), \varphi(x_{i2}), ..., \varphi(x_{i\hat{d}})\} (\hat{d} \geq d) \), a \( \hat{d} \)-by-1 vector containing \( \hat{d} \) features in \( V' \) space, where \( V' \) space is obtained by a mapping function on \( V \) space.

A special case of \( \varphi(x_i) \) is \( \varphi(x_i) = x_i \). In this case, not only training set \( T \) can be linearly separated in \( V \) space, but also we aim to find a linear separation in \( V \) space. However, if \( T \) cannot be linearly separated in \( V \) space, \( \varphi(x_i) \) maps \( x_i \) to a higher dimensional space \( V' \) to get linear separation. In this case, separating boundary is linear in \( V' \) space, i.e., a hyperplane, but nonlinear in \( V \) space. \( w \) is \( d \)-by-1 vector that is normal to the hyperplane in \( V' \) space.

\[ m(x_i) \text{ is a non-negative function that calculates the absolute value of } w^T \varphi(x_i) + b. \] Here we define the margin of the whole training set \( T \):

\[ M = \text{min}\{m(x_1), ..., m(x_i), ..., m(x_n)\}. \]  

Equation 2.9
When $y_i = 1$, $w^T \varphi(x_i) + b > M$ is for training example on the positive side of hyperplane, and $y_i = -1$, $w^T \varphi(x_i) + b < M$ is for training example on the negative side of hyperplane. Compare margins of training examples ($y_i = 1$) with examples ($y_i = -1$) requires us to get absolute value of $w^T \varphi(x_i) + b$.

### 2.2.2 Margin of Training Set

Given a training set $T = \{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_n, y_n)\}$ ($1 \leq i \leq n$), margin of a training set $T$ is the minimum margin for all training examples in $T$:

$$m_T = \min(m(x_1), \ldots, m(x_i), \ldots, m(x_n))$$

Equation 2.10

### 2.3 Solution Region

Solution region is a set of all the solution weight vectors [60]. In geometry, all solution vectors lie inside the solution region. Every training example $(x_i, y_i)$ can define a weight vector $w_i$ and a half-space that is on the positive side of separating hyperplane. Given training set $T$ with $m$ training examples, solution region is the intersection of $m$ half-spaces.

![Figure 2.1 Margin and Solution Region](image_url)
As shown in Figure 2.1 with four training examples $x_1, x_2, x_3, x_4$ with two classes. Blue lines are $x_1, x_2$ and purple lines are $x_3, x_4$. The weight vector $w_1, w_2, w_3, w_4$ is normal to $x_1, x_2, x_3, x_4$, which defines four half-spaces. The intersection of four half-spaces is the solution region for all weight vectors, denoted as region $A$, between vector $w_2$ and $w_3$. Since solution region contains all the possible solution weight vectors $w$ for discriminant functions, it determines all the possible separating hyperplanes that are perpendicular to them. Therefore, solution region $A$ also defines a region $B$, for all possible separating hyperplanes.

2.4 Margin Impact

Overall, margin has an impact on training performance and generalization. Training performance and generalization can be measured by training accuracy and test accuracy. For accuracies, training error and test error (or generalization error) are usually used to represent the loss during training and prediction. The fewer error you make, the larger the accuracy you get.

2.4.1 Training performance

Training performance is directly influenced by margin. Margin is a parameter that changes with the decision boundary. When the decision boundary varies, the training accuracy should change since the separating surface alters. Another issue in choosing training performance is that it influences the generalization. Sometimes, training accuracy can be improved when it is more sensitive to the training data, but it may cause curve fitting and poor generalization. However, good training accuracy can also indicate that the decision boundary may tend to classify complex data with linearly inseparable cases.
2.4.2 Generalization

Generalization provides us an important guidance regarding how to control the margin to prevent overfitting of the classifier. Generalization error is a function that measures the classification error of the discriminant functions on the test data. It is believed that the larger the margin, the less generalization error is obtained. The margin impact on generalization can be analyzed with solution region.

In geometry, when we increase the margin, the solution region \( A \) shrinks, and the region \( B \) shrinks as well. The narrowed solution region imposes the solution weight vectors tend to fall in the “middle” of its region, like weight vector \( a \) in Figure 2.1. In this case, the “middle” of the region \( A \) corresponds to the “middle” of the region \( B \), like separating plane \( H \). Intuitively, all separating hyperplanes lie in the “middle” of region \( B \) is more likely to classify the unseen data correctly. Thus, the generalization error decreases.

2.5 Advantages and Drawbacks of Margin

Still there are some difficulties in applying margin strategy on learning algorithms, which deserves careful study and analysis. First, investigating margin impact on performances requires us to analyze its relationships among other parameters. Sometimes, a large number of parameters determine the performance of learning algorithm together. For instance, soft margin SVM introduces regularization parameter \( C \) to control the trade-off between the size of slack variables and margin, balancing the errors on the SVM training data and margin maximization [62]. Moreover, kernel parameters also influence the data distributions after kernel mapping and affect margin. Thus, these parameters are closely related to margin, so margin impact should be analyzed with other parameters as
wells. Second, some data are inherently complex and even contain noise, and learning from these examples can become a difficulty, let alone the statistical reliable inferences made by the examples. Margin impact on these data may not follow the theoretical and experimental results that we have obtained. Third, a limitation in the choice of training data may lead to poor generalization and over-fitting. In this case, changing margin most likely fails to improve the generalization.

Despite these drawbacks, the promise of the margin-based learning methodology is so encouraging. The significance of margin should be emphasized in today’s machine learning field using big data. One favorable aspect of margin is that discovering margin impact on classifier’s performance can guide us to find an enough good solution quickly. Instead of only using cross-validation to optimize the model parameters, margin can be utilized as an alternative way to adjust the parameters and yield optimal performance. Another attraction is that it can avoid much of laborious design for future classifiers, at the expense of tuning only one parameter: margin.
CHAPTER 3

MARGIN SETTING ALGORITHM

3.1 Training Procedure

Margin setting generates hyperspheres as decision boundaries, called prototypes. Overall, margin setting training procedure can be viewed as two processes: a partition process and an evolution process. They execute iteratively until reaching the stopping condition. Figure 3.1 illustrates the partition process and evolution process of the margin setting algorithm.
3.1.1 Partition Process

The partition process divides the training set into subsets. As one can see, the training process for SVM uses the whole training set to seek hyperplane decision boundaries. These decision boundaries yield the largest margin with only minimal classification errors. However, this becomes difficult to compute when the values of features increase to large amounts. Instead of using the whole training set, margin setting divides the training set into many smaller subsets after partition process in Figure 3.1. It trains hypersphere decision boundaries for those smaller problems with a margin preset.
by the user. There are several advantages to this decomposition method. First, it breaks the whole training sets into some small subsets. Thus, it makes it easier to find decision boundaries for non-separable cases. Second, margin setting also avoids the computation difficulty when facing training sets with a lot of features. Third, a suitable feature selection for classification must be taken into account during the SVM training process, whereas margin setting circumvents this problem.

Specifically, given training set $S$, margin setting generates a partition of the set $S$. Each partition is denoted as the subsets $S_1$, $S_2$, and $S_3$. The training set can be viewed as a union of its subsets. The subsets are non-empty and non-overlapping. In particular, we formulate the partition process in margin setting as:

$$S = \bigcup_{i>1} S_i,$$

where $\emptyset \notin S_i$ and $S_i \cap S_j = \emptyset (i \neq j)$. \hspace{1cm} \text{Equation 3.1}

Margin setting iteratively seeks the prototypes for the subsets. Next, those training points falling into the prototypes are removed from the whole training set. The remaining training sets recursively perform the partition task until no training points are left.

3.1.2 Evolution Process

The evolution process runs iteratively and finds the optimal prototypes using stochastic search. Inspired by the biological evolution approach, the margin setting algorithm performs two steps repeatedly until the stopping conditions are reached. These two steps are constructing initial prototypes and mutating the prototypes. Constructing initial prototypes initially spawns $N$ points randomly in normalized space. These $N$ points are regarded as the centers of prototypes. Second, the distances between each center and each training point are calculated. If this center is the nearest point to the training points
with class $C_p$, it is the center of prototypes with class $C_p$. Third, the radius is the distance from the center to the nearest $C_{p'}$ ($p' \neq p$) class points.

Fourth, we score the constructed initial prototypes using figure of merit. Figure of merit of a prototype is measured by the number of training examples falling into this prototype.

Fifth, we alter the prototypes generated in previous step with randomness and produce the next generation. If the largest figure of merit of current generation is smaller than its direct previous generation, it mutates to the next generation. Otherwise, the evolution process stops, and the partition process begin.

### 3.1.3 Stop Condition

Ideally, the algorithm stops when all the points are partitioned out with an empty reduced set. However, when some points are misclassified, or unclassified, the ideal case may not be reached. Two conditions are used to overcome this obstacle: first, the number of sample points remaining in the reduced set drops to a user-set percent, e.g. 2%, etc. Second, the number of generations is raised to a user-set large limit, e.g. 20, etc. When any of the above two conditions is met, the algorithm is terminated.

### 3.2 An Example of Margin Setting Algorithm

Next, we illustrate margin setting algorithm through a simple example shown below. Suppose two classes of points in 2-dimensional space are shown in Figure 3.2 (a). The red class is denoted as red cross points, and blue class are denoted as blue cross points. Each of them has 12 points. There are total 24 points, and each of them can be denoted as $(X_i, Y_i)$, where $1 \leq i \leq 24$. 
Step 1, we randomly throw $N = 10$ points, denoted as black star points in 2-dimensional space among red class and blue class, shown in Figure 3.2(b).

![Sample Points](image1)

![N Random Points](image2)

Figure 3.2 Margin Setting Algorithm – Generate Random Points

Step 2, generate initial prototypes of the margin setting algorithm, as shown in Figure 3.3(a). Prototypes, i.e., circles in 2-dimensional space, are depicted in two colors, red and blue. Red circles are prototypes for red class points. Blue circles are prototypes for blue class points. Black points are the centers of all prototypes, i.e., circles. If this black point is nearest to the red class points, it is the center of the red circle. Its radius is the distance from this center to the nearest blue class points. On the other hand, if this black point is nearest to the blue class points, it is the center of the blue circle. Then its radius is the distance from this center to the nearest red class points. Record the circle with the largest figure of merit, for red class and blue class, respectively, as shown in Figure 3.3(b). In this case, the figure of merit is 2 for both classes.
Step 3, we can set the margin of the margin setting algorithm by shrinking its radius as shown in Figure 3.4. If the margin is set to 0.2, we should reduce the radius of the circle by 20%. Then, a smaller circle will be stored as the prototypes during algorithm runtime. In this case, the figure of merit remains 2 for both classes.
Figure 3.4 Margin Setting Algorithm – Set Margin

Figure 3.5 Margin Setting Algorithm – Mutation
Step 4, the mutation is performed as shown in Figure 3.5. Take blue class points as an example. There are three circles for blue class points, and their figure of merits is presented in Table 3.1. Recalled that figure of merit is calculated by the number of training examples inside the circle. The blue class points are covered by three circles, denoted by Proto1, Proto2, and Proto3, with a figure of merit 1, 1 and 4, respectively. Therefore, the sum of the figure of merits for all three circles is 6. We can calculate the probability that prototype1 (proto1) is selected by using its figure of merit 1 divided by the total figure merits of all three circles 6, and yield 0.17. Do this for the other two prototypes, Proto2, and Proto3. Then calculate the cumulative probabilities and record in the table. Choose a random number in [0,1], and if this number is 0.3, which is larger than 0.17 and smaller than 0.34, Proto2 is chosen for mutation.

Table 3.1 Margin Setting Algorithm - Figure of Merit for Blue Class

<table>
<thead>
<tr>
<th>Prototpe</th>
<th>Figure of Merit</th>
<th>Probability</th>
<th>Cumulative Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proto1</td>
<td>1</td>
<td>0.17</td>
<td>0.17</td>
</tr>
<tr>
<td>Proto2</td>
<td>1</td>
<td>0.17</td>
<td>0.34</td>
</tr>
<tr>
<td>Proto3</td>
<td>4</td>
<td>0.66</td>
<td>1</td>
</tr>
</tbody>
</table>

How to mutate the center of Proto2? If the center is denoted as \((X_i, Y_i) = (1.65, -3.78)\), then perform the following steps:

1) Choose Random Sign: RS = ±1

2) Calculate Max Perturbation:

\[
X_{Per} = \begin{cases} 
X_i & \text{if } X_i \leq \frac{\min(X_1, \ldots, X_{10}) + \max(X_1, \ldots, X_{10})}{2} \\
\max(X_1, \ldots, X_{10}) - X_i & \text{otherwise.}
\end{cases}
\]
\[ Y_{Per} = \begin{cases} Y_i \text{ if } X_i \leq \frac{\min(Y_1, \ldots, Y_{10}) + \max(Y_1, \ldots, Y_{10})}{2} \\ \max(Y_1, \ldots, Y_{10}) - Y_i \text{ otherwise.} \end{cases} \]  

Equation 3.2

3) Choose Random Number:

\[ \alpha \in [0,1]. \] Finally, the mutated point is denoted as \( (X', Y') \), where:

\[ X' = X_i + RS \times X_{Per} \times \alpha \]

\[ Y' = Y_i + RS \times Y_{Per} \times \alpha \]  

Equation 3.3

(a)  
(b)
Figure 3.6 Margin Setting Algorithm – Prototypes for Each Generation. (a) Prototypes after generation 1; (b) prototypes after generation 2; (c) prototypes after generation 3; (d) prototypes after generation 4; (e) prototypes after generation 5.
Step 5, take the mutated points as input for the random points again, and enter the next mutation. The next mutation begins with step 2. Compare the largest figure of merit between two adjacent mutations. If the largest figure of merit does not increase, stop mutation and save the prototypes for this generation.

Step 6, get a reduced set by excluding the class points inside the prototypes for this generation. Then, start the next generation by returning to step 1. All five generations are needed until all the class points are classified as shown in Figure 3.6.

3.3 Algorithm

The margin setting algorithm is a supervised learning algorithm for multi-class $C_p$ ($p = 1, 2, \ldots P$) pattern recognition. It contains two phases: classification and recognition. In the classification phase, decision boundaries of class $C_p$ are computed from a training set. In the recognition phase, the decision boundaries are used to classify the unlabeled test set. The notation and algorithm are given in the following steps. Classification phase includes step 1) to 8). Step 9) is recognition phase.

Unif (A): uniform distribution on set A.

$\delta$: the magnitude of perturbation.

$\chi$: $\chi$-percent margin

$Q$: largest generation index

$\mu$: a counter that keeps track of the number of mutations

$W$: largest mutation index

$\nu$: a counter that keeps track of the number of generations

Step 1): Construct training set $S$ with $m$ sample points. For each class $C_p$ ($1 \leq p \leq P$), $m_p$ points are selected ($m = m_1 + m_2 + \cdots + m_p$). A training set $S$ consists of
$m$ training samples, $S = \{(x_1, ..., x_m)\}$. Each training sample is an $n$-dimensional vector $x_i = (x_{i1}, x_{i2}, ..., x_{in})$, $1 \leq i \leq m$.

Step 2): Randomly select $N$ n-dimension points in the normalized space $\text{Unif}([0,1])$, and each point is a vector $\omega_k (1 \leq k \leq N)$.

Step 3): Find the center of prototypes with class $C_p$. The Euclidean distance from each $\omega_k$ to each training sample vector $x_i$, and record the minimum value:

$$d_k = \min ||\omega_k - x_i||.$$  
Equation 3.4

Where $\omega_k$ is the center of the class $C_p$, and $x_i$ is with label $C_p$.

Step 4): Find the radius $R_k$ of prototypes with class $C_p$ for each $\omega_k$:

$$R_k = \min ||\omega_k - x_i||.$$  
Equation 3.5

In this case, $x_i$ is with the class label $C_p'$ ($p' \neq p$).

Step 5): Construct prototypes $G_{pi}(i = 1, ..., h)$ of class $C_p$, and compute the figure merit of $G_{pi}$. Figure merit $F_{pi}$ is the number of sample points with class label $C_p$ inside $G_{pi}$. Therefore, $G_{pi}$ with center $\omega_k$, radius $R_k$ and class $C_p$, can be written as $G_{pi} = (\omega_k, R_k, C_p)$. We compute the largest $F_{pi}$ as $LF_p$.

Step 6): For each prototype of class $C_p$ among $G_{pi}(i = 1, ..., h)$, we calculate

$$f_p = \frac{F_{pi}}{\sum F_{pi}}.$$  

Choose a random number $Y$ from $\text{Unif}([0,1])$. Then, pick hypersphere $H_\sigma = (\omega_k, R_k, C_p)$ if subscript $k$ in $\omega_k$ satisfy the following:

$$\sum_{\xi=1}^{k-1} f_\xi < Y \leq \sum_{\xi=1}^{k} f_\xi.$$  
Equation 3.6

Step 7): Mutate $\omega_k$ of each class $C_p$. First, we choose a random sign symbol $\varepsilon$ and pick another number $\alpha$ from $\text{Unif}([0,1])$. $L$ is the maximum perturbation, and $\delta = \varepsilon \alpha L$. The mutated $N$ points are $\omega_k + \delta$. Second, compare the largest figure of merit
between two adjacent generations, if \( LF_{p}^{\mu+1} > LF_{p}^{\mu} \) or \( \mu < Q \). Repeat steps 4 through 6 and mutate again. Otherwise, mutation is stopped and goes to step 8. Then the prototypes of current generation with \( LF_{p}^{\mu} \) are stored, and they are written as \( G = \{ H_{p} = (\omega_{k}, R_{k}, C_{p}) \} \). Once mutation completes, the prototypes for all \( p \) classes is:

\[
\bigcup_{j=1}^{p} (\omega_{k}, R_{k}, C_{p})|_{\mu}.
\]

Equation 3.7

Step 8): Partition to reduced set of class \( C_{p} \) and yield the next generation. First, use prescribed \( \chi \)-percent margin to apply for prototypes \( G \) in step 7. \( R_{k,\chi} = (1 - 0.01\chi)R_{k} \), and \( G' = \{(\omega_{k}, R_{k,\chi}, C_{p})\} \). Second, remove out all sample points in prototype \( G' \), and yield reduced set \( S_{t}' \) for next generation. Third, if \( S_{t} \neq \phi \) or \( v < W \), repeat steps 2 through 8. Otherwise, partition is terminated and store all the prototypes generated in all generations for all classes \( C_{p} (p = 1, 2, ... P) \). Once the partition process completes, the optimal prototypes, i.e. decision boundaries, for all \( p \) classes is

\[
G'' = \bigcup_{i=1}^{p} \bigcup_{j=1}^{p} (\omega_{k}, R_{k}, C_{p})|_{\mu}.
\]

Equation 3.8

Step 9): Recognition. The test set is unlabeled raw data. For all points \( y_{i} \) in \( T \), the Euclidean distance is computed between \( y_{i} \) and \( \omega_{k} \), where \( \omega_{k} \) is the center of prototypes \( G'' \), and if:

\[
||y_{i} - \omega_{k}|| \leq R_{k}.
\]

Equation 3.9

We recognize that \( y_{i} \) is with class label \( C_{p} \), where \( (\omega_{k}, R_{k}, C_{p}) \in G'' \).
CHAPTER 4

ANALYSIS OF MARGIN SETTING ALGORITHM AS A MARGIN-BASED SPHERICAL CLASSIFICATION

4.1 Spherical Classification Analysis

Margin setting algorithm is a margin classifier that generates hyperspheres for classification, called spherical classification. A novel approach that combines margin concept in the support vector machine to spherical classification in margin setting is proposed to analyze margin impact. This analysis aims to explore the impacts of margin chosen on the classification performance.
4.1.1 Performance Metric

Consider the example shown in Figure 4.1. Two classes of points: ten circular points and ten triangular points are linearly non-separable. Let us assume the circle $G_2$ is intended to be the decision boundary for one of the two classes (e.g., circular points). What it implies is that the region outside the red circle belongs to the triangular class. To measure the statistical performance of spherical analysis, we define miss classification probability ($MC$) and over classification probability ($OC$) as follows:

$$MC = \frac{FN}{TP+FN}.$$  \hspace{1cm} \text{Equation 4.1}

$$OC = \frac{FP}{TN+FP}.$$  \hspace{1cm} \text{Equation 4.2}

$TP$: probability of circular points falling within $G_2$;

$FN$: probability of circular points falling outside $G_2$;

$FP$: probability of triangular points falling within $G_2$;

$TN$: probability of triangular points falling outside $G_2$. 

Figure 4.1 Margin Impact on Spherical Classification
The overall performance $OP$ can also be measured by considering $MC$ and $OC$ together:

$$OP = MC + OC.$$  

Equation 4.3

### 4.1.2 Margin Impact

Geometrically, margin is increased by reducing the radius of the hypersphere, which in turn alters the decision boundary. In Figure 4.1, hypersphere decision boundary becomes a circle $G2$ in the 2-D dimensional space, separates the circular points out from the triangular points. Among them, 9 circular points are correctly classified, since they are enclosed by circle $G2$. However, 1 circular point is misclassified, since it is located outside the boundary $G2$. Therefore, $MC$ is 10%. Meanwhile, 3 triangular points fall into $G2$. In this case, they are misclassified as circular points, contributing to 30% $OC$. By adding $MC$ and $OC$ together, the total error probability is 40%.

Margin has a direct impact on the probability of misclassification. $G1$ is obtained by reducing the radius of $G2$ by $\chi$ percent. After the decision boundary shrinks from $G2$ to $G1$, the margin increases from 0 to $\chi$, leading to 2 more circular points outside of boundary $G1$. In this case, $MC$ increases to 30%. But fortunately 3 more triangular points are correctly classified since they are now outside of $G1$, resulting in 0% $OC$. The total error probability $OP$ drops to 30%. Therefore, the margin impact on classification performance for this particular example can be concluded in the following Table 4.1.

<table>
<thead>
<tr>
<th>Margin</th>
<th>$MC$</th>
<th>$OC$</th>
<th>$OP$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increase</td>
<td>Increase</td>
<td>Decrease</td>
<td>Decrease</td>
</tr>
</tbody>
</table>
Moreover, margin also has an impact on the complexity of decision boundaries in spherical classification. For example, in margin setting algorithm, increasing margin yields more hyperspheres to be generated for classification. This effect can be shown in Figure 4.2. It is shown that after enlarging margin from 0.1 to 0.5, the number of circles that classify both red points and green points increases. Besides, most of the circles are small circles in (b) after increasing the margin, while (a) contains more large circles.

Figure 4.2 Increasing Margin Generates More Hyperspheres (circles in 2D space) (a) 0.1 margin; (b) 0.5 margin.

4.2 Experiment

To gain insights into the margin impact on spherical classification in margin setting, experiments were conducted on artificial data in 2-dimensional space. The data contains two classes and it is Gaussian distributed. Its attributes are described in Table 4.2. Our experiments have two stages in analyzing the spherical classification strategies. First, we investigate the results in terms of $MC$, $OC$ and $OP$ for one-sphere classification
Second, increase the number of spheres and get the multi-sphere classification performance in terms of $MC$, $OC$ and $OP$. Since both $MC$ and $OC$ are measured by probability analysis, experiments are carried out using the Monte Carlo method.

<table>
<thead>
<tr>
<th>Gaussian Distribution</th>
<th>Probability Mean</th>
<th>Covariance Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class I</td>
<td>$[-1, 1]$</td>
<td>$\begin{bmatrix} 0.75 &amp; 0 \ 0 &amp; 5 \end{bmatrix}$</td>
</tr>
<tr>
<td>Class II</td>
<td>$[1, -1]$</td>
<td>$\begin{bmatrix} 0.75 &amp; 0 \ 0 &amp; 5 \end{bmatrix}$</td>
</tr>
</tbody>
</table>

We produce the simulated data sets for sphere analysis. As shown in Figure 4.3, the toy data contains two classes: red class and blue class, denoted as red points and blue points respectively. Figure 4.3 (a) presents the one-sphere analysis for classification. Circle for each class data is centered using the mean of the Gaussian distribution, i.e., the center for red circle is $[-1, 1]$ and $[1, -1]$ for blue circle. The radius is set to cover most of the data of its class when the margin is zero. In this case, radius is set to 5. Here, we use a desired $MC$ to control the radius.
Figure 4.3 One-Sphere and Multi-Sphere Classification

As for multi-sphere analysis, more circles are placed in a way that the geometric center of all centers still coincides with the Gaussian mean. Meanwhile, a decrease in radius is used. For simplicity, we set the radius to meet the desired value of $MC$ being less than a very small number that is close to 0, i.e., 0.028. For two circle cases, the centers are $[-1, 2]$ and $[-1, 0]$ for red circles, $[1, 0]$ and $[1, -2]$ for blue circles. The radii are reduced from 5 to 4 to meet the criterion of $MC$. As shown in Figure 4.3 (b) for three circle cases, the toy data contains two classes: red class and blue class, denoted as red points and blue points respectively. For red class points, three circles are with centers $[-1, 3], [-1, 1]$ and $[-1, -1]$. Their geometric centers are located at the Gaussian mean $[-1, 1]$. The radius of three circles is all 3.5. When it comes to four circle cases, the radius is 3.2 and the centers are $[-1, -2], [-1, 0], [-1, 2]$ and $[-1, 4]$ for classifying red class points. The circles for classifying blue the class points are constructed in the same way above.
4.2.1 Probability Analysis & Monte Carlo Methods

Probability analysis is an important way to measure the performance of classification. Spherical classification in margin setting utilizes margin as a strategy to minimize the probability of miss classification and over classification, i.e. $MC$ and $OC$. $MC$ can be obtained from Equation 4.1 and note that $TP + FN = 1$. Hence, $MC = 1 - TP$. For the above multivariate Gaussian distribution in the 2-dimensional space, the random vector $V = [X, Y]$ is with mean $\mu$ and covariance $\Sigma$. $TP$ is the probability of the true positives of one-sphere classification. It can be calculated as a double integral on the probability density function shown as follows:

$$
\iint_D \frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-\rho^2}} e^{-\frac{1}{2(\rho^2-\rho^2)} \left[ \frac{(x-\mu_X)^2}{\sigma_X^2} + \frac{(y-\mu_Y)^2}{\sigma_Y^2} - \frac{2\rho(x-\mu_X)(y-\mu_Y)}{\sigma_X\sigma_Y} \right]} dx dy. \quad \text{Equation 4.4}
$$

The above double integral is evaluated over the area of circle $D$, which is the decision boundary. $D$ is with center $(x_0, y_0)$ and radius $R_0$. It can be represented as follows:

$$(x - x_0)^2 + (y - y_0)^2 = R_0^2. \quad \text{Equation 4.5}$$

Where $\rho$ is the correlation between $X, Y$. $\mu$ and $\Sigma$ are:

$$
\mu = \left( \begin{array}{c} \mu_X \\ \mu_Y \end{array} \right), \quad \Sigma = \left( \begin{array}{cc} \sigma_X^2 & \rho \sigma_X \sigma_Y \\ \rho \sigma_X \sigma_Y & \sigma_Y^2 \end{array} \right). \quad \text{Equation 4.6}
$$

As for multi-sphere classification, double integral is taken from the areas of total $m$ ($m \geq 1$) circles. Each circle is denoted as one region $D_j (j = 1, 2, ..., m)$, and the set of all the regions $D_\ell$ is:

$$D_\ell = \sum_{j=1}^{m} D_j. \quad \text{Equation 4.7}$$

To evaluate the numerical values of the double integrals in Equation 4.4, we use Monte Carlo method due to lack of closed-form expressions. Monte Carlo methods
implement stochastic simulation that was first applied to model diffusion of neutrons through fissile materials in 1947. In 1970, it was formally defined as a method utilizing a random sequence of numbers to form a population sample, which represents the solution of a problem as a parameter of statistical estimation [63]. One important example of Monte Carlo methods is stochastic integration, also called Monte Carlo integration. Specifically, it deals with the case when the integral boundaries are complicated, and integration is multidimensional. It evaluates a definite integral through a set of observations that simulate the target value.

In our experiment, Monte Carlo integration is utilized. It repeatedly produces pseudorandom data to get the numerical estimation of the probability. To evaluate the numerical values of the double integrals in Equation 4.4, we run Monte Carlo experiments and obtained the $MC$ and $OC$ results.

4.2.2 One-sphere Analysis

One-sphere analysis studies the fundamental margin impact on one sphere. Here one-sphere strategy generates one sphere for each class, as a spherical classification in margin setting. It can be seen from Figure 4.3 that the red circle serves as the classification boundary covering the red class points while the blue circle is the decision boundary enclosing the blue class points. In this case, still some red points and blue points fall outside of all decision boundaries, leaving them unclassified. We will analyze its effect later.
Figure 4.4 One-Sphere Analysis for Miss Classification and Over Classification

Figure 4.5 One-Sphere Analysis for Overall Performance

The $MC$ and $OC$ performance is shown in Figure 4.4. The results clearly show a steady increase of $MC$ and a decrease of $OC$ when margin increases. Figure 4.5 shows that $MC + OC$ value goes down to the minimum when the margin is increased to 0.55.
Then it rises again and quickly reaches to the maximum. Overall, the results show a similar margin impact on spherical classification in Table 4.1. Moreover, the results show that the overall probability of error \((MC + OC)\) could be minimized by choosing a certain margin. In other words, in order to achieve a low probability of error, the margin cannot be too large, not too small either. This explains that margin setting is not a maximal margin classifier, such as SVM and some other algorithms [77-81]. Margin setting can be viewed as an optimal margin classifier.

### 4.2.3 Multi-Sphere Analysis

Multi-sphere analysis concentrates on the margin impact of multiple spheres. Multiple spheres are obtained by adding more small spheres to classify each class instead of using some large sphere as shown in previous Figure 4.2. This analysis provides insight into the theoretical foundation of the margin setting algorithm that utilizes multi-sphere as decision boundaries. Multi-sphere boundaries have the ability to classify the linear non-separable training set with high accuracy. Specifically, margin setting breaks the training set \(T\) into several small subsets \(T_1, T_2, T_3, \ldots, T_n (n > 1)\), and \(T\) can be viewed as the union of these subsets:

\[
T = T_1 \cup T_2 \cup T_3 \ldots T_n.
\]  

Equation 4.8
One of the advantages of using for multiple spheres is that subsets are easier to classify and distinguish among one another. Instead of using some large hyperspheres for the whole training data $T$, margin setting generates one small hypersphere shows for one subset $T_i (i = 1, \ldots, n)$ in each generation. As illustrated in Figure 4.6, each subset $T_i$ is finally enclosed by one hypersphere $HS_i (i = 1, \ldots, n)$. When the margin setting algorithm is terminated after the stopping condition is met, the multiple hyperspheres generated for all generations are the decision boundaries for the training set.
Another benefit of using multiple spheres is that both $MC$ and $OC$ are lower than the case of one sphere for a large margin within a certain range, as can be seen in Figure
4.7 and Figure 4.8. Within a margin range of (0, 0.7), $MC$ declines when the number of circles increases. For example, $MC$ for one/two/three/four circles are 0.301, 0.231, 0.163, 0.131 when margin equals to 0.5. However, when margin exceeds 0.7, the case of four circles slightly produces more errors than the case of three circles. It means that a very large margin may degrade the performance of multiple spheres. However, it should be noted that not only the number of spheres, but also the locations of multiple spheres affects the performance. Note here that the analysis is based on the strategy that 4 circles are evenly located around the Gaussian means, similar to the 3-circle and 2-circle cases. If we move the 4 circles around, we might be able to lower the overall probabilities of error. This would be an interesting direction of further investigation. It also explains why the margin setting algorithm, as a multi-sphere learning algorithm, attempts to employ a stochastic scheme to find the optimal locations of spheres as well. On the other hand, it can be seen in Figure 4.8 that the probability of over classification is reduced by using more spheres for a wide range of margin from 0 to 1. Therefore, the overall performance is improved for multiple spheres.
It is also worth noting that the performance of the optimal margin is better for multiple circles. Optimal margin is obtained when the $MC + OC$ reaches a minimum. For an example shown in Figure 4.9, the optimal margins for one/two/three/four circles are all 0.55. At the same time, the $MC + OC$ value are 0.58, 0.47, 0.40 and 0.36. That is, the more circles, the better the minimum overall probability of error. Besides, the overall trend for $MC$, $OC$ and $MC + OC$ in multi-sphere analysis conforms to earlier results for one-sphere analysis.

4.2.4 Non-Classification Analysis

The margin setting algorithm, which makes use of multi-spherical classification, might leave some data not classified (i.e., the non-classified data did not fall within any of the spheres). Large number of unclassified data is undesirable and affects the overall performance as well as $MC$ and $OC$. On the other hand, leaving a small number of data samples mis-classified or non-classified would get better generalization performance.
It can be seen that the probability of non-classification increases with margin in Figure 4.10. Recall that the margin reaches optimum when it rises to 0.55. For this optimal margin, non-classification reaches 11.8% for the Four-Circle case. On the other hand, it can be also seen that more circles contribute to a smaller probability of non-classification. Non-classification goes down dramatically from 36.3% to 11.8% for 0.55 margin.

4.3 Summary

This chapter presents a novel approach that combines the margin concepts in SVM to spherical classification in the margin setting algorithm. We analyze the margin impact on spherical classification in the margin setting algorithm. Our analysis provides insights into the performance of this algorithm as a spherical classifier. It explains why the overall probability of error can be lowered by increasing the margin within a certain range. Moreover, our analysis based on Gaussian distributed data shows that using...
multiple spheres as decision boundaries can improve the classification performance of the single sphere classifier. Therefore, it explains in part the success of the margin setting algorithm, which employs a large number of spheres in classifying real data. Apart from the simulated data sets, spherical classification of margin setting has been successfully applied to some benchmark data sets from UCI machine learning repository in the next chapter.
CHAPTER 5

IMPACT OF SETTING MARGIN ON MARIN SETTING ALGORITHM AND
SUPPORT VECTOR MACHINES

5.1 Motivation

Both margin setting and support vector machine are motivated by solving pattern recognition problems. Pattern recognition in machine learning is a method of making statistical inferences from the perceptual data. One important example of pattern recognition is pattern classification. The main purpose of classification is to understand and perceive the discriminant among patterns [62].

In particular, a pattern is defined as a pair of \( \langle x, y \rangle \), where \( x \) is a feature vector composed of a collection of features, \( y \) is the class label. Features can be a series of attributes or properties of data, which can be numerical (i.e., length), or symbolic (i.e., shape). If the feature vector \( x \) has \( m \) features, it is in \( m \)-dimension feature space. Note that examples \( \langle x_i, y_i \rangle \) of a pattern, shares similar features values may belong to the same class, and if it has different features, they should be categorized into different classes. Therefore, it is desirable that a good feature vector contains features that can discriminate among one another very well. A binary classification in Figure 5.1 shows two classes of
training examples, denoted as triangle points and circle points. In Figure 5.1(a), we can easily find a linear classifier to distinguish them and the samples has good feature vectors, and they are linearly separable. However, sometimes we cannot find a linear classifier to classify the two classes in Figure 5.1(b) and they contain bad feature vectors. Hence, they are linearly non-separable.

Figure 5.1 Decision Boundary of Margin Setting and SVM (a) Linearly Separable by SVM; (b) Linearly Non-separable by Margin Setting.
In general, the motivation of margin setting and SVM is generating classifiers to classify different classes of data. However, these two algorithms are developed under different considerations. First, the original design of margin setting classifier tackles difficult non-separable scenarios, but support vector machine is motivated by linear discriminant functions, with a linear separation as shown in Figure 5.1 (a). Second, as for non-separable scenario, support vector machine obtains a linear separation by mapping the input space into a higher dimensional space. However, margin setting only considers classifying the training data in its original feature space with an optimal classifier.

5.2 Decision Boundary

A decision boundary is a hypersurface that separates the vector space into two subsets. In pattern classification, both margin setting and SVM find decision boundary that partitions two classes of training examples. The difference is that the decision boundary of SVM is a hyperplane, while margin setting employs a hypersphere. The binary classification problem is shown in Figure 5.1 to illustrate the difference. Figure 5.1(a) presents that decision boundary of SVM is a line when the two classes of training examples are linearly separable. However, if the two classes of training examples are linearly non-separable, we may need to map the original two-dimension space to a higher dimensional space. Thus, the separating line is generalized to a separating hyperplane. It is seen from Figure 5.1(b) that margin setting decision boundary is presented by red circles and blue circles. Training examples in red circles are classified as one class, and blue circles cover examples for another class. Another difference is that during the testing phase for classifying unknown labeled data, i.e. test data, some test data falls outside of
hyperspheres. They are called unclassified data. However, there is no unclassified data for SVM classification, and it classifies all test data.

5.2.1 Decision boundary of SVM

Hyperplane: support vector machine decision boundary is a hyperplane \( H: (w, b) \), where \( w \) is a normal vector, or a weight vector, perpendicular to the hyperplane with initial value \( w_0 = 0 \). It is adjusted iteratively each time when training examples are misclassified by current \( w \). \( b \) is intercept or bias. All the training examples \( x_i \) on hyperplane \( H \) satisfy

\[
    w^T x_i + b = 0. \tag{5.1}
\]

To assign class labels to each class for test data, we use another two hyperplane \( H1 \) and \( H2 \) to determine their labels, satisfying the equations follows:

\[
\begin{align*}
    H1: & \quad w^T x_i + b \geq 1, \quad \text{if } y_i = 1 \\
    H2: & \quad w^T x_i + b \leq -1, \quad \text{if } y_i = -1
\end{align*} \tag{5.2}
\]

![Figure 5.2 Hyperplane and Margins of SVM](image)
Support vector machine introduces \( H1 \) and \( H2 \) as shown in Figure 5.2 (a). For hard margin case, no training examples are expected between \( H1 \) and \( H2 \). Therefore, SVM desires to find a wider separation between these two classes of training examples. The decision boundary in two-dimensional spaces is line \( H \). \( H1 \) and \( H2 \) are parallel to \( H \). The training examples on the \( H1 \) and \( H2 \) are called support vectors (SVs) [16]. There are two basic characteristics of it. First, they are the closest training examples to the decision boundary. Second, if we alter the support vectors, the decision boundary need to be changed.

**5.2.2 Decision boundary of Margin Setting**

Prototype: Margin setting decision boundary is hyperspheres are called prototypes, which are defined as center-radius forms:

\[
G = \{ (\delta_k, R_k, C_i), 1 \leq k \leq N, i = 1, 2, ..., P \}.
\]

Equation 5.3

Where \( \delta_k \) is the center of \( G \), \( R_k \) is the radius of \( G \), and \( C_i \) is the class label. \( m \) and \( k \) are natural numbers. \( N \) is the number of prototypes belonging to class \( C_i \).

In order to assign class labels to each class for test data, decision boundary should satisfy the equations of Euclidean distance from \( x_i \) to \( \delta_k \):

\[
||x_i - \delta_k|| \leq R_k.
\]

Equation 5.4

Where \( x_i \) is test data from test set \( S = \{ x_1, x_2, ..., x_i, ... \} \). \( \delta_k \) is the center of \( G \). \( R_k \) is the radius of prototypes. \( x_i \) is assigned with class label \( C_i \) only when above equation holds.
5.3 Margin

Decision boundary allows us to assign training examples to their proper classes. However, decision boundaries for both support vector machine and margin setting are not unique. How can we find the optimal decision boundary? A crucial metric, margin is considered in determining the optimum of the decision boundary. The most favorable decision boundary classifies training examples with a very confident set of predictions. Margin in the margin-based algorithm is defined as a non-negative real-valued function, and its magnitude is associated with confidence in the prediction. Besides, a good confidence of prediction yields to good generalization of the statistical margin classifier. Thus, the magnitude of margin can improve the generalization ability.

5.3.1 Margin of SVM

Margin of support vector machine can be defined in two notions: function margin and geometric margin. Both of them are associated with the confidence of the prediction since margin changes with the decision boundary. Recall that the typical target of statistical learning algorithms is to find a classifier that satisfies two conditions: 1) a good confidence of prediction, 2) minimal expected loss. Corresponding to the above two conditions, support vector machine aims to find a decision boundary that has 1) the largest margin and 2) minimal classification errors. Therefore, support vector machine is not only a margin-based statistical learning algorithm, but also an optimal and maximal margin classifier.

A. Algebraic Interpretation of Margin

Definition 5.2 (Function Margin of SVM) Given a training example \((x_i, y_i)\), function margin of \((x_i, y_i)\) with respect to a hyperplane \((w, b)\) is to be the quantity
\[ \gamma(x_i) = y_i(w^T x_i + b). \]  

Equation 5.5

\( \gamma(x_i) \) is a function that calculates the value of \( w^T x_i + b \) for binary classification, where \( y_i = \{1,-1\} \). \( \gamma(x_i) \) is positive if and only if the training example is correctly classified. One drawback in function margin is that it does not give accurate measure of its magnitude. \( \gamma(x_i) \) can be arbitrary large if we scale up \( w \) and \( b \) without changing the classifier, since there are no constraints of \( w \). Hence, geometric margin is introduced to set constraints of \( w \) to overcome the drawbacks of function margin.

The function margin of SVM indicates that a large margin contributes to the high confidence prediction. Therefore, SVM is a large margin classifier [71-72]. Intuitively, \( \gamma(x_i) > 0 \) yields a large margin. In this case, \((w^T x_i + b)\) and label \( y_i \) share the same sign, leading to the correct prediction. Therefore, large margin requires correct prediction. On the contrary, false prediction causes \((w^T x_i + b)\) and label \( y_i \) have different sign. In this case, \( \gamma(x_i) < 0 \), only small negative margin can be obtained.

B. Geometric Interpretation of Margin

Definition 5.3 (Geometric Margin of SVM) Given a training example \((x_i, y_i)\), the geometric margin of \((x_i, y_i)\) with respect to a hyperplane \((w, b)\) algebraic distance \( d \) to be the quantity

\[ d = \left| \frac{\gamma(x_i)}{\|w\|} \right|. \]  

Equation 5.6

Where \( \|w\| \) is the Euclidean norm of \( w \). Specifically, if \( w \) equals to a unit vector, the geometric margin equals to function margin.

Geometric margin of a training example is defined as the Euclidean distance from the point \((x_i, y_i)\) to the decision hyperplane. It depicts the width of separation between support vectors of different classes of training examples. To illustrate it, we consider a
simple linearly separable case in two dimensional spaces in Figure 5.2 and find the geometric margin of point A \((x_i, y_i)\). Point B is on the decision hyperplane. Line segment \(AB\) is perpendicular to hyperplane. Geometric margin \(d\) is defined as the distance from point A to the hyperplane, i.e., the length of segment \(AB\). Recall that \(w\) is a normal vector that is perpendicular to the hyperplane as well. Since the unit vector \(w/\|w\|\) and point B can be represented as \(x_i - d \cdot w/\|w\|\), and point B is on the hyperplane satisfying \(w^T x_i + b = 0\), we have the following equation and it yields \(d\) in Equation 5.6:

\[
 w^T (x_i - d \frac{w}{\|w\|}) + b = 0. 
\]  

Equation 5.7

Support vector machine is an optimal margin classifier, and one important goal is to maximize the geometric margin of the all its hyperplanes and find the maximal one. In this case, SVM is also a maximal margin classifier. The maximal margin is called the geometric margin of the given training set. In what follows, we present the derived definitions of geometric margins in support vector machine:

Margin of hyperplane: Given the training set \(T\), margin of hyperplane \((w, b)\) is defined as the smallest geometric margins of all its training examples

\[
 \gamma_{(w,b)} = \min(\gamma(x_1), \ldots, \gamma(x_i), \ldots, \gamma(x_m))
\]  

Equation 5.8

Margin of a training set: Given all separating hyperplanes \(\{(w_1, b_1), \ldots, (w_n, b_n)\}\) and its corresponding geometric margin are \(\{\gamma_{w_1, b_1}, \ldots, \gamma_{w_n, b_n}\}\). Margin of a training set \(T\) is defined as the maximum geometric margin of all separating hyperplanes

\[
 \gamma_T = \max (\gamma_{(w_1, b_1)}, \ldots, \gamma_{(w_n, b_n)})
\]  

Equation 5.9

Support vector machine maximizes the geometric margin of its hyperplanes. When \(y_i = 1\), \(w^T \varphi(x_i) + b > \gamma_T\) is for training example on the positive side of
hyperplane, and \( y_i = -1 \), \( w^T \phi(x_i) + b < \gamma_T \) is for training example on the negative side of hyperplane.

Margin of the hyperplanes is the distance from support vectors to the hyperplane as shown in Figure 5.2, namely the distance between \( H1 \) and \( H2 \). Let \( d_+ \) and \( d_- \) be the shortest distance from the separating hyperplane to the support vectors of class with label (+1) and label (-1) respectively. Margin is quantitatively calculated as:

\[
\text{Margin} = d_+ + (d_-) = \frac{2}{\|w\|}
\]

Equation 5.10

To maximize it, we need to minimize \( \|w\| \) to maximize the margin. We also define margin area is the region between hyperplanes of support vectors. In Figure 5.2, the region between \( H1 \) and \( H2 \) is the margin area. Maximizing the margin also enlarges the margin area.

5.3.2 Margin of Margin Setting

Similar to SVM, margin of the margin setting changes its decision boundary. The decision boundary in margin setting is a hypersphere called a prototype. When the radius of prototypes shrinks, margin varies. Therefore, in geometry, margin measures the difference volume between the prototypes of the training sets before and after shrinking their radii.
Figure 5.3 Hypersphere and Margins of Margin Setting

Geometrically, margin is computed as the volume of the region between the two concentric hyperspheres. Since distance varies, the volume of the region varies correspondingly. The two concentric hyperspheres consist of 1) the hypersphere that does not shrink the radius, called zero margin hypersphere, and 2) the hypersphere that reduces the radius of the zero margin hypersphere by χ-percent. It measures the magnitude of margin and is defined as follows:

χ-percent margin: Given a training set $T$, and the generated prototypes $(\delta_k, R_{k,0}, C_i)$ for class $C_i$, where $k \geq 1$ and $i \geq 2$. The corresponding radius of zero margin is $R_{k,0}$, and the radius of χ margin is the quantity

$$R_{k,\chi} = (1 - \chi)R_{k,0}$$

where $0 \leq \chi < 100$  

Equation 5.11
Definition 5.4 (Geometric Margin of Margin Setting) Given a training set $T$, the generated prototypes $(\delta_k, R_{k,0}, C_i)$ for class $C_i$ ($k \geq 1$ and $i \geq 2$), and the radius of $\chi$-percent margin $R_{k,\chi}$, we define the geometric margin of $T$ as the sum of the volumes of the regions between concentric $n$-spheres of zero margins and $\chi$-percent margins with respect to a class $C_i$. For a multi-classification of $m$ classes, each class $C_i$ generates $\theta_i$ prototypes. The geometric margin is to be the quantity $\gamma_s$

$$\gamma_s = \left\{ \begin{array}{ll}
\bigcup_{t=1}^{m} \bigcup_{k=1}^{t} \frac{\pi^t}{t!} (R_{k,t,\chi}^{2t} - R_{k,t,0}^{2t}) & \text{if } m = 2t \\
\bigcup_{t=1}^{m} \bigcup_{k=1}^{2(t+1)} \frac{\pi^t}{(2t+1)!!} (R_{k,t,\chi}^{2t+1} - R_{k,t,0}^{2t+1}) & \text{if } m = 2t + 1
\end{array} \right.$$  

Equation 5.12

where $t!$ is a factorial and $(2t + 1)!!$ is a double factorial. Specifically, $(2t + 1)!!$ is calculated as the product of even integers less than or equal to $(2t + 1)$ but greater than or equal to 2.

To illustrate the margin definition, let us consider a simple binary classification problem shown in Figure 5.3. Training set $T$ is distinguished by circular points, i.e., class $C_1$, and triangular points represents the remaining class $C_2$. Each class has ten training points. Margin setting breaks the twenty training points into four groups of subsets and finds prototypes for each of them. In particular, the prototypes in two dimensional spaces are in the shape of circles. The circular points are enclosed by two different prototypes, i.e., red circles, and the triangular points are surrounded by another two blue circles. Next, we analyze the prototype with largest radius in the top left of the figure, which is a large red circle denoted as prototype $G_1$. Specifically, $G_1$ contains six class $C_1$ training points with center $\delta_0$ and radius $R_0$, and it can be denoted as $G_1 = \{\delta_0, R_0, C_1\}$. Inside of $G_1$, a smaller green dashed circle that contains only two training points is represented as prototype $G_2 = \{\delta_0, R_1, C_1\}$. Clearly, $G_1$ and $G_2$ are concentric circles. To obtain $G_2$, we
only shrink the radius of $G_2$ from $R_0$ to $R_1$. Moreover, the gray region between $G_1$ and $G_2$ is defined as the margin. Note that $G_1$ is the prototypes called zero-margin. The prototypes whose radii’ have been shrunk, such as $G_2$, have $\chi$-percent margin. It is obtained by shrinking the radius from $R_0$ to $R_1$ by $\chi$ percent.

5.4 Margin Impact

Margin has an impact on training performance and generalization. Training performance and generalization can be measured by training accuracy and test accuracy. Regarding the accuracies, training error and test error (or generalization error) are usually used to represent the loss during training and prediction [76]. The fewer errors made, the larger the accuracy obtained. Several researchers discussed the generalization bounds of SVM and other classifiers. However, they do not explicitly concentrate on the impact of margin on generalization [83-86]. In this section, we analyze the margin impacts on these two algorithms specifically and discuss their differences.

5.4.1 Margin Impact of SVM

SVM has three essential features: optimal hyperplane, kernel tricks, and soft margin. To analyze the margin impact, we first examine margin relations to optimal hyperplane and “kernel tricks”, and then extend our margin impacts on soft margin for the linearly non-separable case. In this case, margin influences training error and generalization error.

First, SVM explicitly finds an optimal separating hyperplane with the maximized margin. To maximize the margin, weight vector $\|w\|$ should be minimized. The minimization problem focuses only on the training examples that are difficult to classify, i.e., support vectors, instead of the entire training set or misclassified training examples.
Therefore, an equivalent dual problem of minimizing $\|w\|$ is a maximization problem solving by QP (Quadratic Programming) below:

$$\text{maximize } w(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$

subject to $\sum_{i=1}^{m} y_i \alpha_i = 0$

$0 \leq \alpha_i \leq C, \ i = 1, \ldots, m.$  \hspace{1cm} \text{Equation 5.13}$

Where $\alpha_1, \ldots, \alpha_m$ is the Lagrangian multiplier associated with each training example $(x_i, y_i)$. The Lagrangian multipliers are bounded by $C$, called a box constraint, and $\alpha_i$ is the lagrangian multiplier for support vectors. It can be solved by the sequential minimal optimization (SMO) algorithm [64]. When $\alpha_i = 0$, it is associated with training examples that are far beyond the separating plane. They are not support vectors and they do not appear in the maximization problem above. When $\alpha_i = C$, it is associated with training examples that are missclassified within the margin area, called the bounded support vectors (bounded SVs). In this case, training examples are linearly non-separable.

When $0 < \alpha_i < C$, the training examples lie on the boundary of the margin area, called the free support vectors (free SVs) [73]. When $0 < \alpha_i < C$, free support vectors that lies on $H1$ and $H2$ as shown in previously Figure 5.2. These support vectors, only a part of the training data, actually decide the separating plane that lies in the middle of the margin area, and determine the margin as well.

Maximizing margin leads to the optimal hyperplane. The optimal hyperplane improves the generalization error. It is also proved that larger margins on the training set result in a superior upper bound on the generalization error [35].
Figure 5.4 The Smallest Hypersphere Enclosing Two Classes of Training Set

Second, linearly non-separable training examples can obtain linearly separation through kernel mapping. It preprocesses the linear non-separable training examples by mapping them from the original input space to a typically higher dimensional space. In the higher dimensional space, let $R$ be the radius of the smallest hypersphere that encloses all training examples shown in Figure 5.4, then the margin’s impact on generalization errors can be derived by margin/radius bound [66-70]:

$$\gamma_T = \frac{R^2}{\sqrt{IE}}.$$  \hspace{1cm} \text{Equation 5.14}

Where $E$ is the number of errors for leave-one-out cross-validation, and $l$ is the size of the training set, and $\gamma_T$ is the margin of training set that is defined before. The above equation indicates that the magnitude of margin is bounded by kernel type and errors. Given that a specific kernel, $R$ is a constant. An increase in margin $\gamma_T$ results in a smaller $E$. Therefore, the above equation implies that when kernel factors are equal,
higher margin leads to lower errors. In order to minimize $E$, SVM needs to maximize its margin.

Third, “soft margin” is introduced to classify the non-linearly separable examples [82]. In this case, the maximized margin separating hyperplane may not be the optimal one. Instead, a tradeoff between maximized margin and tolerance of noise is considered. A regularization parameter $C$ is added to measure the tolerance. This tolerance also balances large margin and small margin. Although a lot of researchers have worked on how to tune the parameters of SVM to get better performance, they fail to investigate algorithm performance with respect to margin [66, 67, 74, 75]. The non-negative slack variables $\varepsilon$ presents the margin violation constraints. Therefore, a formation of the problem is presented below:

$$
\begin{align*}
\text{minimize} & \quad \frac{1}{2}||w|| + C \sum_{i=1}^{m} \varepsilon_i \\
\text{subject to} & \quad y_i(w^T x_i + b) \geq 1 - \varepsilon_i, \\
& \quad \varepsilon_i > 0, \; i = 1, \ldots, m.
\end{align*}
$$

Equation 5.15

Training errors and generalization errors can be reduced with the magnitude of margin. In the case of soft margin, the margin effect relates to regularization parameter $C$. When we increase the margin, the minimization problem above attempts to make a wider separation between different classes. Increasing the margin requires a smaller $C$, which tolerates more misclassified examples that are inside margin area. In this case, more training errors occur but fewer generalization errors are made. In addition, increasing margin brings more support vectors due to the fact that it tends to turn more training examples into bounded support vectors. To explain it, the following equation holds:
\[ SVs = FSVs + BSVs \]  

Equation 5.16

Where \( SVs \) is the total number of support vectors; \( FSVs \) is the number of free support vectors; \( BSVs \) is the number of bounded support vectors. Note that \( SVs \) is less than or equal to the total number of training examples. When we increase margin, \( BSVs \) increases but \( FSVs \) may be not. However, if \( SVs \) increases to the number of training examples, enlarging margin causes more \( BSVs \) and fewer \( FSVs \).

On the other hand, when we reduce the margin, a tight and narrow separation is obtained by putting more weight on the slack variables \( \varepsilon_i \). More weight on slack variables is achieved by increasing \( C \), leading to less toleration of noises. As a result, we make fewer training errors and possibly more generalization errors.

### 5.4.2 Margin Impact of Margin Setting

More training error occurs when we increase margin. As shown in previous Figure 5.3, from prototype \( G_1 \) to prototype \( G_2 \), margin is increased from 0 to a gray region with the area of \( \pi(R_0^2 - R_1^2) \) after shrinking the radius size from \( R_0 \) to \( R_1 \). At the same time, the number of training points inside of \( G_1 \) is reduced from 6 to only 2 in prototype \( G_2 \), which turns out 4 training points are unclassified. In this example, we can see that enlarging the margin may result in an increase of training classification error.

Generalization error is reduced when we slightly increase margin. Generalization error occurs when using the generated hypersphere classifier to classify the unlabeled data, which is the recognition phase of margin setting. In most cases, margin setting favors a large margin rather than zero-margin. The reason is that a large margin can indicate a more confident prediction of our classification than small margin. A larger margin also contributes to leaving more room for unseen data without incurring
classification errors. Another effect of large margin is to improve generalization errors [87-88]. For instance, as shown in Figure 5.3, the top left prototype $G_1$ includes class $C_1$ and the top right prototype $G_3$ encloses another class $C_2$. $G_1$ and $G_3$ are classifiers for binary classification problem of two classes $C_1$ and $C_2$. We increase the margin of $G_1$ by shrinking its radius to get $G_2$. Likewise, $G_4$ is obtained by enlarging the zero-margin of $G_3$. Intuitively, instead of using $G_1$ and $G_3$, $G_2$ and $G_4$ are considered as our decision boundary, which contributes to larger margins of classes $C_1$ and $C_2$. Enlarging the margin tends to generate smaller circles like $G_3$ and $G_4$. In this case, testing examples, i.e., unseen data, fall into smaller circles with less generalization error than large circles. Large circles are more likely to cover more noise data than smaller circles.

Until now, we have discussed and theoretically analyzed the margin impacts of margin setting and SVM. The conclusion is provided in Table 5.1 below.

Table 5.1 Theoretical Analysis of Margin Impact on Training and Generalization

<table>
<thead>
<tr>
<th>Increase Margin</th>
<th>SVM</th>
<th>Margin Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Accuracy</td>
<td>decrease</td>
<td>decrease</td>
</tr>
<tr>
<td>Generalization</td>
<td>increase</td>
<td>increase slightly, then decrease</td>
</tr>
</tbody>
</table>

5.5 Experiment

We compare the margin impact of support vector machine and margin setting for pattern classification through artificial toy data sets and benchmark data sets from UCI repository in LIBSVM format [92-94]. In particular, we illustrate how margin impacts training error and generalization error.
5.5.1 Methodology

A. Parameters and model selection

The toy data set is randomly generated by Gaussian distribution. User specified parameters are set by default parameters. In margin setting, parameters were chosen as follows: maximum generation $MG = 20$, maximum mutation $MM = 20$, and number of random points $NR = 20$. Then $\chi$-percent margin is set by the user to enlarge margin. LIBSVM default parameters are used for SVM classification. The radial basis function (RBF) kernel is chosen and regularization parameter $C$ is varied to change SVM margin. The default kernel parameter $\gamma$ is set as $\gamma = 1/\text{num\_features}$. Num\_features is the number of features of the training set.

Benchmark data sets are more complicated than toy data sets, so the user specified parameters are set slightly differently. In margin setting, maximum generation $MG = 100$, maximum mutation $MM = 20$, and number of random points $NR = 100$. In addition, linear, polynomial and RBF kernel are all used for SVM with default parameters in LIBSVM. The training set is randomly selected as one-third of the data. The remaining two-thirds are for testing. Categorical attributes of the data sets are scaled to the range $[0, 1]$. Each experiment is repeated 10 times and the accuracy is obtained by averaging the results.

B. Margin range

Different from the margin range that is used in [61], the range of margin we plotted is $(0, 1)$. This comes from the fact that the margin is a positive value that is calculated via the correctly classified training examples only. These training examples are support vectors in SVM. No negative value of margin is considered for incorrectly
classified training examples. On the other hand, margin setting's margin is less than 1, i.e., $\chi < 100\%$. Although the margin of SVM can exceed 1, we choose a common range that SVM and margin setting for comparison purpose.

### 5.5.2 Toy Data Set

The 2-D toy data set is randomly generated by two Gaussian distribution. It has two classes, and Table 5.2 describes its attributes. In this experiment, we randomly generated 500 red points for class I and 500 green points for class II. In order to explore the margin impact on the toy dataset, we conduct the experiment as follows: the data set is randomly split into a training set and a testing set. The latter containing 95% of the entries to make this problem more challenging.

<table>
<thead>
<tr>
<th>Gaussian Distribution</th>
<th>Probability Mean</th>
<th>Covariance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class I</td>
<td>[-1 1]</td>
<td>[0.75;0;0.5]</td>
</tr>
<tr>
<td>Class II</td>
<td>[1 -1]</td>
<td>[0.75;0;0.5]</td>
</tr>
</tbody>
</table>

Table 5.2 Attributes of Toy Dataset
Figure 5.5 Classification Results of Toy data as a Function of Margin (a) 0 Margin of Margin Setting; (b) 0.04 Margin of Margin Setting; (c) 0.5 Margin of Margin Setting; (d) 0 Margin of SVM; (e) 0.3 Margin of SVM; (f) 0.6 Margin of SVM.

To visualize the margin impacts on generalization performance, we draw the altered decision boundary as we increase margin in Figure 5.5. Margin setting generates more circles from (a) to (c) while SVM changes from curves in (d) to a nearly straight line with a small curve in (f). As for margin setting, more circles classify more test data correctly in (b), for both red points and green points. However, if margin is sufficiently large, such as a value of 0.5 in (c), the number of non-classified points dramatically increases with margin. In this case, all the non-classification is counted as errors and accuracy goes down. As for SVM, zero margin in (d) tries to achieve an extremely low error on the training set but brings more errors on test data. The increased margin in (f) tolerates more errors on the training set, but more test data are classified by the line with
much less stationary points. In this case, generalization performance is improved with margin.

![Graphs showing training accuracy and generalization as a function of margin.](a). (b)

Figure 5.6 Training accuracy, Generalization as a Function of Margin for Toy Data Set (a) Margin impact on training accuracy of SVM “SVM_Tr” and test accuracy of SVM “SVM_Tt”; and (b) Margin impact on training accuracy of Margin Setting “MS_Tr”, as well as test accuracy of Margin Setting “MS_Tt”.

To further quantitatively measure the margin impact on the training errors and generalization errors, we plotted them as a function of margin in Figure 5.6 for margin setting and SVM. Figure 5.6 (b) shows that the generalization error of margin setting is improved for a small non-zero margin within a certain range [0, 0.04]. However, above a size 0.04 margin, the accuracy for testing set declines. This result reveals that better performance can be obtained only when we slightly enlarge margin for margin setting. On the other hand, it is seen from Figure 5.6(a) that the testing accuracy of SVM begins to increase from 0.1 margin to 0.5 margin. Within the range of 0.1 margin to 0.5 margin, performance remains unchanged due to the fact that the separating plane almost does not
change. In this case, the number of misclassified examples does not change due to the fact that the number of bounded support vectors keeps unchanged. The number of bounded support vectors is close to the total number of support vectors, and very few free support vectors are left. The fewer free support vectors, the fewer the changes of the separating plane. Either slightly enlarging margin or significantly enlarging margin in high-dimensional space may not alter the separating plane too much or enhance performances noticeably. In comparison, margin setting enlarges margin in the original feature space without kernel mapping. It is evident that testing accuracy drops for large margin with small circles. Small circles are likely to drive out correctly classified data. Besides, it is noteworthy that margin setting achieves better generalization performance than SVM for a small range of margin magnitude, specifically from 0 to 0.18.

We also observe that training performance of margin setting is always superior to SVM with an increased margin in Figure 5.6. This fact reveals that it is easier for margin setting to generate a boundary that classifies the linearly non-separable training set correctly. Margin setting maintains a 0% training error and a higher generalization performance than SVM without apparent overtraining for margin less than 0.15. We expect that training performance will be degraded with a very large margin. However, it is surprising that 100% training accuracy can always be achieved even up to a 0.9 margin. However, SVM tends to decrease training accuracy while improving testing accuracies with margin. Note that with a margin larger than 0.5, training performance remains almost unchanged to avoid overtraining.
5.5.3 Benchmark Data Sets

We further test the margin impact on several publicly available benchmark data sets from UCI Repository. The data sets are selected with multiple types. They vary from small size, low dimensional data to large size, high dimensional data. Both binary classification and multi-class classification datasets are chosen. For each data set, we randomly choose approximately one-third of the data for constructing training set. The remaining data is used for testing. We report the results that how classification error and generalization error are influenced by margin. The characteristics of these datasets are listed in Table 5.3.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># training</th>
<th># testing</th>
<th># features</th>
<th># classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liver Disorders</td>
<td>116</td>
<td>229</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Indians Diabetes</td>
<td>257</td>
<td>511</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>228</td>
<td>455</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>117</td>
<td>234</td>
<td>34</td>
<td>2</td>
</tr>
<tr>
<td>Mushrooms</td>
<td>2709</td>
<td>5415</td>
<td>112</td>
<td>2</td>
</tr>
<tr>
<td>Iris</td>
<td>50</td>
<td>100</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Glass</td>
<td>74</td>
<td>140</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>Segment</td>
<td>770</td>
<td>1540</td>
<td>19</td>
<td>7</td>
</tr>
</tbody>
</table>
Figure 5.7 Margin Impact on Training Accuracy for Benchmark Data Sets (a) Training accuracy as a function of margin for SVM linear kernel, (b) polynomial kernel, (c) RBF kernel; (d) Training accuracy as a function or margin for margin setting.

Margin impacts on training performance of benchmark data are reported in Figure 5.7. Due to the effect of radius/margin bounds for generalization, we analyze the margin impacts of SVM for linear kernel, polynomial kernel, and RBF kernel separately. Overall, the training accuracy decreases with margin for all models in (a), (b), and (c). Note that during a wide range of margin, SVM maintains a minimum accuracy after margin is
increased to a small value. For instance, the Glass data set keeps the training accuracy of 35.14% above 0.2 margin in (a), 0.02 margin in (b), and 0.08 margin in (c). The Iris data set retains the accuracy of 90.20% after 0.3 margin for linear kernel, an accuracy of 68.63% after 0.2 margin for polynomial, and an accuracy of 90.20% after 0.2 margin for RBF kernel. This trend indicates that SVM is more sensitive to margin impact for a small margin, while margin setting’s training performance degrades in a full range of margin. This fact also explains why SVM gains a noticeable advantage through kernel mapping.

Instead of classifying data in the original space as margin setting, SVM separates data which tends to be easier to classify in higher dimensional space. Higher dimensional space changes the data distribution so that the number of bounded support vectors increases to the number the total number of support vectors, making the accuracy stay unchanged. However, margin setting tends to achieve better training accuracy than SVM for a small range of margin, as shown in Figure 5.7. In other words, the training performance of SVM drops more quickly than margin setting. For example, the Breast Cancer dataset trained by margin setting yields an accuracy higher than both SVM RBF kernel and polynomial kernel with a wide margin range of [0, 0.5].
Margin impact on the generalization with benchmark data is illustrated in Figure 5.8. It is clear that test accuracy is increased with a small margin range from 0 to around 0.03 for margin setting in (d) for all benchmark data sets except Liver Disorder. It is also shown in (c) of RBF kernel, the test performance of seven data sets when the margin is less than 0.05, adding further evidence to our theoretical analysis that generalization is
improved with margin. However, this margin boosting generalization impact is not as conspicuous for SVM for linear kernel and polynomial results in (a) and (b). This may be attributed to the fact that some data are too noisy, even after they are mapped to higher dimensional space [95]. Therefore, margin impact on improving generalization is shown only within a certain range. Tuning margin in margin setting within a small range can definitely yield better generalization performance.

5.6 Summary

Margin is an important factor that affects the generalization for all margin-based algorithms. We explore the margin impact of a new learning algorithm, margin setting, by comparing with the support vector machine. Margin impacts are analyzed and discussed in terms of training performance and generalization performance theoretically. The theoretical analysis compares the margin definitions and margin impact between margin setting and SVM. The margin definitions are discussed in functional and geometric representations. The margin impact of margin setting is discussed by presenting how margin change the decision boundary and performance. As for SVM, margin altering decision boundary and impacting performance are analyzed with considerations of kernel selection and soft margin. Extensive experiments are carried out on toy data sets and benchmark data sets. Experimental results demonstrate that training performance tends to decline with an increasing margin. However, generalization performance improves when we increase margin within a certain range. The results successfully justifies our theoretical analysis and additionally indicates that tune margin in a small range can certainly increase the generalization performance of margin setting. In the future, our work can be extended to the margin impacts on the computational and
spatial complexity of these two algorithms. Another future work includes comparing and discussing margin impacts on other popular margin classifiers, such as AdaBoost and Perceptron [90-91].
CHAPTER 6

LEARNING-BASED SWITCHING MEDIAN FILTER FOR NOISE REMOVAL

In this section, a novel application of the supervised learning algorithm, margin setting, is proposed for noise detection. Second, a new noise-free two-stage filtering scheme is proposed for restoring the corrupted image for a broad range of noise densities. The impulse noise suppression performance of margin setting is compared and analyzed with SVM and Neural Networks.

6.1 Learning-based Noise Detection

Accurate noise detection is a significant factor that contributes to the satisfactory performance of impulse noise suppression. Current switching median filter removes the noise but produces many false positives for images with low noise densities. It detects many clean pixels as noise pixels, resulting in a high over detection rate [51]. Moreover, the filter scheme after noise detection impacts the final performance of impulse noise suppression as well. We propose a learning-based switching filter, MS-NFTS. This filter utilizes 1) a supervised learning algorithm, margin setting for noise detection, and 2) NFTS filtering for image reconstruction. Noise detection performs the “switching” strategy which distinguishes the noise pixels from non-noise pixels. After noise detection,
a noise decision map is built to identify the noise positions. Later, a noise free two-stage filter is proposed to estimate the original pixel and restore the image. The flowchart of MS-NFTS filter is shown in Figure 6.1.

![Flowchart](image)

Figure 6.1 MS-NFTS Impulse Noise Suppression

Color images chosen for impulse suppression are common color images in RGB space. A color vector \( (O_{i,j}^R, O_{i,j}^G, O_{i,j}^B) \) can represent the pixel at location \((i, j)\) in RGB color space. In particular, \( O_{i,j}^R \) denotes the red (R) component, and \( O_{i,j}^G \) and \( O_{i,j}^B \) are the green (G) and blue (B) component, respectively. When a color image is corrupted with salt-and-pepper noise, the R, G, and B components are injected with noise randomly and independently. The original pixel is denoted as \( o_{ij} = (O_{i,j}^R, O_{i,j}^G, O_{i,j}^B) \) and the corrupted pixel is denoted as \( n_{ij} = (N_{i,j}^R, N_{i,j}^G, N_{i,j}^B) \). For a color image contaminated with noise density \( p \), the R component of the noise pixel satisfies the following noise model (G, B component follow the same model) [49]:

\[
P(x) = \begin{cases} 
\frac{p}{2} & \text{for } N_{i,j}^R = 0 \\
1 - p & \text{for } N_{i,j}^R = O_{i,j}^R \\
\frac{p}{2} & \text{for } N_{i,j}^R = 255
\end{cases}
\]

Equation 6.1
\( P(x) \) denotes the percentage of entire image pixels. For example, if \( p = 0.2 \), approximately 20% of the image pixels are corrupted with salt-and-pepper noise, and half of them are salt noise. The remaining 80% of the image pixels are not corrupted.

### 6.2 Noise-Free Two-Stage Filtering

Although an effective noise detection method with high accuracy mainly contributes to the performance of impulse suppression, the filtering scheme after detection can have a significant impact on the overall quality of the restored image as well. The noise free two-stage filter (NFTS) proposed in this section is a simple two-stage median filter that can achieve substantially better performance than well-accepted impulse noise filters in highly corrupted salt-and-pepper noise images.

NFTS is modified based on the filtering scheme proposed by Fabijanska et al. [54]. Our modification includes two aspects: First, instead of using a varying window size for different noise density ranges, a fixed 3 \( \times \) 3 window size is used in the first stage filtering. Second, for very high noise density (> 60%) images, the method in [54] fails to deal with the case that no noise-free pixels exist in window, as shown in Figure 6.2. Hence, our method is to label this center pixel with intensity value 0 and use bigger windows, up to a 21 \( \times \) 21 window size to filter this pixel.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>0</td>
<td>255</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>255</td>
<td>255</td>
</tr>
</tbody>
</table>

Figure 6.2 No Noise-free Pixels Scenario
The flowchart of the NFTS filter is shown in Figure 6.3. The filtering process is also carried out separately in three components: $N_{i,j}^R, N_{i,j}^G$ and $N_{i,j}^B$ of the noise image. The detailed steps are summarized after the flowchart.

![Flowchart of NFTS filter](image)

**Figure 6.3 Flowchart of NFTS filter**

Step 1) Given the noise image $N$, a two-dimensional binary decision map $B$ is built after noise detection. $B$ is the same size as $N$. Each entry of $B$, denoted as $B(x,y)$, corresponds to each pixel $N(x,y)$ in the noise image, where $x$, $y$ is the
coordinate of the pixel. If \( B(x,y) = 1 \), then \( N(x,y) \) is the pixel corrupted with noise. If \( B(x,y) = 0 \), then \( N(x,y) \) is the noise-free pixel.

\[
\begin{array}{ccc}
N(x-1, y-1) & N(x,y-1) & N(x+1, y-1) \\
N(x-1, y) & N(x,y) & N(x+1, y) \\
N(x-1, y+1) & N(x,y+1) & N(x+1, y+1)
\end{array}
\]

Figure 6.4 A 3 × 3 Filtering Window

Step 2) First stage filtering: impose a 3 × 3 window, as shown in Fig.6.4. The current pixel \( N(x,y) \) is the center pixel. Pad zeroes on image \( N \) to deal with pixels on the borders. Examine the value of \( B(x,y) \) to decide whether \( N(x,y) \) is a noise pixel, as described in step 1.

Step 3) If \( N(x,y) \) is a noise pixel, examine its neighborhood pixels and form a set \( W_D \) (\( D = 3 \)):

\[
W_D = \{ N(x-1,y-1), N(x,y-1), N(x+1,y-1), N(x-1,y), N(x+1,y), N(x-1,y+1), N(x,y+1), N(x+1,y+1) \}.
\]

Equation 6.2

Decide whether each pixel in \( W \) is a noise pixel or noise-free pixel by examining its corresponding pixel in decision map \( B \). If the number of noise-free pixels \( K > 0 \), then go to the next step. If the number of noise-free pixels \( K = 0 \), add a label on this pixel and slide the window to the next position. Then, go back to
step 2. If \( N(x, y) \) is a noise-free pixel, the value of \( N(x, y) \) is unchanged.

Step 4) Replace the value of \( N(x, y) \) with the median of noise-free pixels in \( W \). Slide the window to the next position and go back to step 2). If you finish filtering the last pixel in \( N \), go to step 5).

Step 5) Second stage filtering: impose a \( 5 \times 5 \) window and filter the labeled pixels remaining in the first stage filtering.

Step 6) If the number of noise-free pixels in the neighborhood \( K' > 0 \), replace the value of \( N(x, y) \) with the median of the noise-free pixel in \( W_D \) (\( D = 5, 7, 9 \ldots 21 \)). If \( K' = 0 \), keep the label unchanged and slide the window to the next labeled pixel.

Step 7) After filtering the last pixel in \( N \), impose a \( 7 \times 7 \) window and go back to step 6) again. The next time step 7 is executed, impose a \( 9 \times 9 \) window, then an \( 11 \times 11 \), and then a \( 13 \times 13 \ldots \) and so on until you reach a \( 21 \times 21 \) window.

Step 8) Consider computational complexity, filtering is terminated for the size \( 21 \times 21 \) window. As for the remaining labeled pixels, their values are kept unchanged.

6.3 Experiment

The proposed filter (NFTS-MS) utilizes margin setting (MS) for noise detection and a noise-free two-stage (NFTS) filter to restore the image. Margin setting impulse suppression algorithm is tested with color images that are corrupted with salt-and-pepper noise. Without loss of generality, typical standard test images [96] are used for experiment, as shown in Figure 6.5. They are the six 24-bit RGB images Cornfield, Boats, Fruits, Goldhill, Lena, and Yacht with image resolutions of \( 480 \times 512, 787 \times 576, 512 \times 480, 720 \times 576, 512 \times 512 \) and \( 512 \times 480 \) pixels, respectively. The experiment contains two parts: detection accuracy measurement and fidelity
measurement. We add salt-and-pepper noise with a wide range of noise density, from 5% to 95%.

Figure 6.5 Test Images (a) Airplane, (b) Boats, (c) Fruits, (d) Goldhill, (e) Lena, (f) Yacht.
Table 6.1 Methods Compared in Experiment

<table>
<thead>
<tr>
<th>Method</th>
<th>Noise Detection</th>
<th>Filtering Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVMS</td>
<td>SVM</td>
<td>SMF</td>
</tr>
<tr>
<td>NNS</td>
<td>NN</td>
<td>SMF</td>
</tr>
<tr>
<td>MSS</td>
<td>MS</td>
<td>SMF</td>
</tr>
<tr>
<td>SVMN</td>
<td>SVM</td>
<td>NFTS</td>
</tr>
<tr>
<td>NNN</td>
<td>NN</td>
<td>NFTS</td>
</tr>
<tr>
<td>MSN</td>
<td>MS</td>
<td>NFTS</td>
</tr>
</tbody>
</table>

In this experiment, 40 training points are randomly selected from noise images. Among them, 20 points are noise pixels and the other 20 points are non-noise pixels. The proposed MS-NFTS is compared with six other methods shown in Table 6.1. In particular, margin setting is compared with two other popular supervised learning methods, neural network and support vector machine. Margin setting is implemented in MATLAB® 2012a version without tuning its parameters. The other two methods also use the default parameters in the same environment. Feed-forward back-propagation network is used for implementation of the neural network.

6.3.1 Noise Detection Analysis

Noise detection analysis includes detection accuracy presented by means of miss detection rate (MD%) and over detection rate (OD%) as follows:

\[ MD\% = \frac{FN}{TP+FN} \]

Equation 6.3

\[ OD\% = \frac{FP}{TN+FP} \]

Equation 6.4
Where $TP$ and $FP$ are the true positive and false positive for noise pixels. $TN$ and $FN$ are the true negative and false negatives for noise pixels. In particular, the true positives rate of noise is an indicator for detecting all the true noise pixels as noise. False positive recognizes the clean pixels as noise.

Table 6.2 Noise Detection Performance of Margin Setting Compared with Neural Network and SVM. The test images are (a) “Boats” (b) “Lena” (c) “Yacht”.

(a) Color image *Boats*

<table>
<thead>
<tr>
<th>Noise Ratio (%)</th>
<th>Miss Detection Rate (MD %)</th>
<th>Over Detection Rate (OD %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SVM</td>
<td>NN</td>
</tr>
<tr>
<td>5</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>10</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>20</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>30</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>40</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>50</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>60</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>70</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>80</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>90</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>95</td>
<td>0.07</td>
<td>0.07</td>
</tr>
</tbody>
</table>
(b) Color image *Lena*

<table>
<thead>
<tr>
<th>Noise Ratio (%)</th>
<th>Miss Detection Rate (MD %)</th>
<th>Over Detection Rate (OD %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SVM</td>
<td>NN</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>70</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>80</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>95</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(c) Color image *Yacht*

<table>
<thead>
<tr>
<th>Noise Ratio (%)</th>
<th>Miss Detection Rate (MD %)</th>
<th>Over Detection Rate (OD %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SVM</td>
<td>NN</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Margin (%)</td>
<td>MD%</td>
<td>OD%</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| 10        | 9.402          | 5.843          | 0.264  
| 20        | 8.828          | 5.855          | 0.250  
| 30        | 8.171          | 5.864          | 0.230  
| 40        | 7.460          | 5.882          | 0.210  
| 50        | 6.236          | 5.876          | 0.193  
| 60        | 5.669          | 5.873          | 0.168  
| 70        | 4.560          | 5.890          | 0.136  
| 80        | 3.290          | 5.923          | 0.104  
| 90        | 1.798          | 6.058          | 0.07   
| 95        | 0.965          | 6.307          | 0.049  

Table 6.2 presents the results of MD% and OD% for three test images: Boats, Lena, and Yacht. Margin setting is compared with Neural Network (NN) and SVM. The three test images are corrupted with noise densities varying from 5% to 95%. For miss detection performance, the results show that MS outperforms SVM and NN with less MD%. For example, as for MD% of image Boats, SVM and NN outputs 0.07% more than MS for all noise densities in average. For over detection performance, MS again yields much less OD% than NN and SVM for all cases. The OD% decreases when the noise density goes down for MS and SVM. However, OD% increases for NN.

The above results indicate that: 1) Margin setting produces less generalization error than neural network and support vector machine for default parameter values. In particular, for margin setting, the maximum generation $MG=20$. Maximum mutation $MM=20$ and zero margin are used. For neural network, a feed forward back-propagation network is implemented. Other parameters are set with default values, such as the
maximum number of epochs to train set to 10, performance goal set to 0, and learning rate set to 0.01. For support vector machine, RBF kernel is used. Other parameters are set for default values. 2) Theoretically, both margin setting and support vector machine converge towards the best solution. However, neural networks use a heuristic approach which cannot guarantee the best solution. Therefore, when noise density is higher, more pixels are corrupted with noise and reduce the chances of over detection.

### 6.3.2 Filter Scheme Analysis

Filter scheme analysis includes the comparison of impulse suppression performance among six filters listed in Table 6.1. The performance is quantified by four popular evaluation metrics. They are mean square error (MSE), peak signal-to-noise ratio (PSNR), image enhancement factor (IEF) and structural SIMilarity (SSIM) index.

Image Enhancement Factor (IEF) is a measurement of image quality. It is the ratio of mean square error before noise-reduction to the mean square error after noise-reduction. The noise-reduction process can be viewed as a filter, and IEF qualitatively indicates the quality improvement. The mathematical formula of MSE, PSNR and IEF are given as follows [49]:

\[
MSE = \frac{\sum_{c=r,g,b} \sum_{i=1}^{M} \sum_{j=1}^{N} [d(i,j) - o(i,j)]^2}{3 \times MN}.
\]  

\[
PSNR = 10 \log_{10} \left( \frac{l^2}{MSE} \right).
\]  

\[
IEF = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} [n(i,j) - o(i,j)]^2}{\sum_{i=1}^{M} \sum_{j=1}^{N} [d(i,j) - o(i,j)]^2}.
\]  

In the above formulas, \( n(i, j) \) is the pixel intensity at location \((i, j)\) in the noisy image, \( o(i, j) \) is the pixel intensity at location \((i, j)\) in the original image, and \( d(i, j) \) is the pixel intensity at location \((i, j)\) in the denoised image. \( M \times N \) is the dimension of the...
images in pixels. L is the intensity level of the used gray scale standard test image. 255 is chosen for calculation.

Usually, a lower MSE and a higher PSNR and IEF indicate that the denoised image is with higher quality. However, sometimes it may not be the case. They have been proven to fail to evaluate the image qualities adequately. Images have the same MSE in regard to the original standard non-noise image, but sometimes their perceptual qualities have drastically differences [101-102]. Hence, SSIM index is considered in our experiment for comparison. The value of SSIM ranges from 0 to 1. The original non-noise image has a value of 1, and if the restored image has a value close to 1, it has a better image quality. SSIM calculates the similarity assessment in terms of luminance \( l(x, y) \), contrast \( c(x, y) \) and structures \( s(x, y) \). The equation of SSIM is as follows:

\[
SSIM(x, y) = [l(x, y)]^\alpha[c(x, y)]^\beta[s(x, y)]^\gamma.
\]  
Equation 6.8

In the above formula, \( l(x, y) \), \( c(x, y) \), and \( s(x, y) \) are given by the following three equations:

\[
l(x, y) = \frac{2\mu_x\mu_y+C_1}{\mu_x^2+\mu_y^2+C_1},
\]  
Equation 6.9

\[
c(x, y) = \frac{2\sigma_x\sigma_y+C_2}{\sigma_x^2+\sigma_y^2+C_2},
\]  
Equation 6.10

\[
s(x, y) = \frac{\sigma_{xy}+C_3}{\sigma_x\sigma_y+C_3}.
\]  
Equation 6.11

In the above formulas, \( \mu_x \) and \( \sigma_x \) are the mean intensities and standard deviation of the denoised image, respectively. \( \mu_y \) and \( \sigma_y \) are the mean intensities and standard deviation of the original image, respectively. \( \sigma_{xy} \) is the cross-correlation between the original and denoised image. \( C_1 = (K_1L)^2 \) and \( C_2 = (K_2L)^2 \), where \( K_1, K_2 \ll 1 \). By
default, \( K_1 = 0.01, K_2 = 0.03, C_3 = C_2/2. \) \( \alpha = \beta = \gamma = 1, \) and \( L \) is the intensity levels equals to 255.
Table 6.3 Impulse Noise Suppression Performance of MSN filter Compared with Other Methods. Test images are (a) “Fruits”; (b) “Goldhill”; (c) “Lena”; (d) “Yacht”.

(a) Color image Fruits

<table>
<thead>
<tr>
<th>Filters</th>
<th>20%</th>
<th></th>
<th></th>
<th>50%</th>
<th></th>
<th></th>
<th>70%</th>
<th></th>
<th></th>
<th>95%</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSE</td>
<td>PSNR</td>
<td>IEF</td>
<td>SSIM</td>
<td>MSE</td>
<td>PSNR</td>
<td>IEF</td>
<td>SSIM</td>
<td>MSE</td>
<td>PSNR</td>
<td>IEF</td>
<td>SSIM</td>
</tr>
<tr>
<td>SVMS</td>
<td>26.51</td>
<td>41.01</td>
<td>161.34</td>
<td>0.978</td>
<td>206.3</td>
<td>37.41</td>
<td>51.72</td>
<td>0.835</td>
<td>1126.1</td>
<td>32.87</td>
<td>13.27</td>
<td>0.449</td>
</tr>
<tr>
<td>NNS</td>
<td>16.62</td>
<td>42.03</td>
<td>257.34</td>
<td>0.989</td>
<td>187.0</td>
<td>36.77</td>
<td>57.07</td>
<td>0.858</td>
<td>1055.4</td>
<td>33.01</td>
<td>14.16</td>
<td>0.473</td>
</tr>
<tr>
<td>MSS</td>
<td>15.57</td>
<td>42.17</td>
<td>274.77</td>
<td>0.990</td>
<td>183.8</td>
<td>36.81</td>
<td>58.06</td>
<td>0.861</td>
<td>1040.8</td>
<td>33.04</td>
<td>14.36</td>
<td>0.478</td>
</tr>
<tr>
<td>SVMN</td>
<td>273.55</td>
<td>35.95</td>
<td>15.61</td>
<td>0.812</td>
<td>328.8</td>
<td>35.55</td>
<td>32.42</td>
<td>0.784</td>
<td>396.53</td>
<td>35.14</td>
<td>37.66</td>
<td>0.746</td>
</tr>
<tr>
<td>NNN</td>
<td>46.11</td>
<td>39.81</td>
<td>92.63</td>
<td>0.957</td>
<td>86.23</td>
<td>38.45</td>
<td>123.71</td>
<td>0.933</td>
<td>136.18</td>
<td>37.46</td>
<td>109.66</td>
<td>0.899</td>
</tr>
<tr>
<td>MSN</td>
<td><strong>12.45</strong></td>
<td><strong>42.65</strong></td>
<td><strong>343.21</strong></td>
<td><strong>0.990</strong></td>
<td><strong>50.65</strong></td>
<td><strong>39.61</strong></td>
<td><strong>210.72</strong></td>
<td><strong>0.966</strong></td>
<td><strong>99.58</strong></td>
<td><strong>38.14</strong></td>
<td><strong>149.99</strong></td>
<td><strong>0.931</strong></td>
</tr>
</tbody>
</table>

(b) Color image Goldhill

<table>
<thead>
<tr>
<th>Filters</th>
<th>20%</th>
<th></th>
<th></th>
<th>50%</th>
<th></th>
<th></th>
<th>70%</th>
<th></th>
<th></th>
<th>95%</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSE</td>
<td>PSNR</td>
<td>IEF</td>
<td>SSIM</td>
<td>MSE</td>
<td>PSNR</td>
<td>IEF</td>
<td>SSIM</td>
<td>MSE</td>
<td>PSNR</td>
<td>IEF</td>
<td>SSIM</td>
</tr>
<tr>
<td>SVMS</td>
<td>32.34</td>
<td>40.58</td>
<td>127.33</td>
<td>0.974</td>
<td>217.01</td>
<td>36.45</td>
<td>47.33</td>
<td>0.840</td>
<td>1147.4</td>
<td>32.83</td>
<td>12.54</td>
<td>0.480</td>
</tr>
<tr>
<td>Filters</td>
<td>MSE</td>
<td>PSNR</td>
<td>IEF</td>
<td>SSIM</td>
<td>MSE</td>
<td>PSNR</td>
<td>IEF</td>
<td>SSIM</td>
<td>MSE</td>
<td>PSNR</td>
<td>IEF</td>
<td>SSIM</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>SVMS</td>
<td>29.03</td>
<td>40.82</td>
<td>136.07</td>
<td>0.980</td>
<td>235.27</td>
<td>36.27</td>
<td>42.02</td>
<td>0.851</td>
<td>1127.2</td>
<td>32.87</td>
<td>12.26</td>
<td>0.489</td>
</tr>
<tr>
<td>NNS</td>
<td>21.93</td>
<td>41.43</td>
<td>180.14</td>
<td>0.986</td>
<td>207.27</td>
<td>36.55</td>
<td>47.70</td>
<td>0.860</td>
<td>1054.8</td>
<td>33.01</td>
<td>13.10</td>
<td>0.501</td>
</tr>
<tr>
<td>MSS</td>
<td>21.68</td>
<td>41.45</td>
<td>182.21</td>
<td>0.986</td>
<td>205.89</td>
<td>36.56</td>
<td>48.02</td>
<td>0.899</td>
<td>1052.1</td>
<td>33.02</td>
<td>13.14</td>
<td>0.502</td>
</tr>
<tr>
<td>SVMN</td>
<td>405.83</td>
<td>35.09</td>
<td>9.73</td>
<td>0.920</td>
<td>512.76</td>
<td>34.58</td>
<td>19.28</td>
<td>0.886</td>
<td>678.61</td>
<td>33.97</td>
<td>20.36</td>
<td>0.845</td>
</tr>
<tr>
<td>NNN</td>
<td>26.44</td>
<td>41.02</td>
<td>149.46</td>
<td>0.988</td>
<td>93.27</td>
<td>38.28</td>
<td>106.03</td>
<td>0.959</td>
<td>175.44</td>
<td>36.91</td>
<td>78.78</td>
<td>0.920</td>
</tr>
<tr>
<td>MSN</td>
<td>15.37</td>
<td>42.20</td>
<td>257.18</td>
<td>0.989</td>
<td>80.46</td>
<td>38.60</td>
<td>122.92</td>
<td>0.960</td>
<td>161.26</td>
<td>37.09</td>
<td>85.71</td>
<td>0.924</td>
</tr>
</tbody>
</table>

(c) Color image *Lena*
(d) Color image Yacht

<table>
<thead>
<tr>
<th>Filters</th>
<th>20%</th>
<th>50%</th>
<th>70%</th>
<th>95%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSE</td>
<td>PSNR</td>
<td>IEF</td>
<td>SSIM</td>
</tr>
<tr>
<td>SVMS</td>
<td>47.16</td>
<td>39.76</td>
<td>85.17</td>
<td>0.983</td>
</tr>
<tr>
<td>NNS</td>
<td>33.85</td>
<td>40.48</td>
<td>118.67</td>
<td>0.990</td>
</tr>
<tr>
<td>MSS</td>
<td>31.71</td>
<td>40.62</td>
<td>126.67</td>
<td>0.991</td>
</tr>
<tr>
<td>SVMN</td>
<td>282.08</td>
<td>35.88</td>
<td>14.21</td>
<td>0.890</td>
</tr>
<tr>
<td>NNN</td>
<td>75.66</td>
<td>38.74</td>
<td>53.10</td>
<td>0.978</td>
</tr>
<tr>
<td>MSN</td>
<td>29.10</td>
<td>40.81</td>
<td>137.89</td>
<td>0.993</td>
</tr>
</tbody>
</table>

96
Table 6.3 presents the results of experiments for the suppression of impulse salt-and-pepper noise on RGB test images Fruits, Goldhill, Lena, and Yacht. The performances of other supervised learning-based filters, SVMS, NNS, MSS, SVMN, and NNN are used for comparison. It is shown that MSN consistently restores images with the lowest MSN and highest PSNR, IEF, and SSIM for all test images with noise densities from 20% to an extremely high 95%. In addition, MSN filter produces much better results than the SVM filter, as well as the NNN filter. For example, in the 50% noise density of image Lena, the IEF values of SVMN, NNN, and MSN are 19.28, 106.03 and 122.92. In the same noise density of image Yacht, the MSE values are 397.85, 175.16 and 113.00. The reason is that during noise detection, MS outputs the lowest over detection rate among NN and SVM reported in Table I, with only 0.01% for image Lena, and 0.19% for image Yacht, while NN and SVM both have a relative high over detection rate. On the other hand, by comparing MSS with MSN, it is obvious that the NFTS filter achieves better performance than SMF filter for noise densities from 20% to 95%. For example, for the PSNR values of the image Fruits, MSS outputs only 0.48 dB more than MSN for 20% noise density. When noise density reaches 95%, the PSNR of MSS is 7.45 dB more than MSN, indicating a nearly 16 times performance increase. In addition, for image Goldhill, the differences of PSNR rise from 0.64 dB to 7.02 dB. However, NFTS fails to perform better than SMF when SVMS is compared with SVMN for noise density less than 70%, as well as NNS is compared with NNN for noise density less than 50%. It is because, during noise detection, SVM and NN perform much worse than MS at low noise densities.
Figure 6.6 Restored Image using MSN filter Compared with Other Filters (a) noise image “Lena” is corrupted with 60% noise density; (b) SMF output; (c) SVMS output; (d) NNS output; (e) MSS output; (f) SVMN output; (g) NNN output; (h) MSN output.

Figure 6.6 shows the visual quality of the restored image Lena, which is corrupted with 60% impulse noise density in (a). Visually, the image quality is not very sensitive among (c) SVMS, (d) NNS and (e) MMS. The reason is the SMF filter schemes they use are not robust. However, the image superiority of (h) is easily seen, compared with (f) using SVMN and (g) using NNN. Therefore, the MSN filter again yields the best performance among other filters in visualization results.
Figure 6.7 Impulse Suppression Performance of MSN filter Compared with Other Filters for Color Image “Cornfield” (a) PSNR; (b) IEF; (c) SSIM. Color image “Boats” (d) PSNR; (e) IEF; (f) SSIM.

Figure 6.7 shows the impulse suppression performance results for the other two test images: Cornfield and Boats. It is evident that MSN achieves the higher PSNR and IEF for noise densities larger than 30%, not 20% like the other four images: Fruits, Goldhill, Lena and Yacht in TABLE 6.3. Note that when the noise density is less than 20%, MSN does not perform better than the MSS filter. There are two reasons. First, the noise detection performance impacts the performance of the NFTS filter. During noise detection, the MD% and OD% of images Cornfield and Boats are worse than the other four images. In this case, the superiority of the MSN filter to the MSS filter is delayed. The advantages can only be revealed with a slightly higher noise density, from 20% to 30%. Similarly, the superiority of SVMN to SVMS is delayed to be shown when noise density is larger than 60%. Second, when the noise density is low (<30%), the over detection rate is higher than that of high noise densities. Thus, more non-noise pixels are prone to be treated as noise pixels. Because the NFTS filter replaces the pixels with the median of non-noise pixels, some non-noise pixels (treated as noise pixels) are excluded, leading to information loss during restoration. However, this side effect can be alleviated with less over detection when noise densities become higher. Moreover, comparing the performance of the SVMN and NNN filter in (a) and (b), it is shown that the NNN filter outputs worse PSNR values than the SVMN filter in (a). However, the NNN filter performs better than SVMN in (b). It is because the noise detection performance of
neural network is better for image Boats but worse for image Cornfield. In addition, we can see that MSN filter gains higher SSIM values for a wide range of noise densities.

![Figures](image1.png)

Figure 6.8 Restored Image using MSN Filter Compared with Other Filters. The test image “Lena”, “Fruits”, “Goldhill” are all corrupted with 95% impulse noise (a) Lena noise image; (b) Lena SVMN output; (c) Lena NNN output; (d) Lena MSN output; (e) Fruits noise image; (f) Fruits SVMN output; (g) Fruits NNN output; (h) Fruits MSN output; (i) Goldhill noise image; (j) Goldhill MSS output; (k) Goldhill SVMN output; (l) Goldhill MSN output.
Figure 6.8 shows the visualization results of restored images with the extremely high noise density (95%). The test images are Lena, Fruits, and Goldhill. The left most column, which consists of (a), (e), (i), has images corrupted with 95% noise density. The first row contains results for Lena images. Apparently, MSN filter output in (d) achieves better performance than NNN output in (c) and SVMN output in (b). Some details and edges are not preserved in (b) and (c) with the appearance of wrong colored pixels. The Fruits image in the second row also indicates the robust and superior performance of the MSN filter output in (h). Some unexpected color pixels blur the images details in (f) and (g). The last row shows that the MSN output in (l) is far better than the MSS output in (j). The SVMN output in (k) removes many image details because of a much higher over detection rate during noise detection compared to the over detection rate of margin setting.

6.4 Summary

In this chapter, a novel learning-based switching median filter, called MSN, is proposed for impulse noise suppression from corrupted color images. The prominent feature of the new filter is the supervised learning-based algorithm, margin setting, for noise detection. Another new feature is a new noise-free two-stage filter scheme to reconstruct the images. Extensive experiments are conducted for six standard natural color images for noise densities ranging from 5% to extremely as high as 95%. MSN filters are compared to five other learning-based filters based on neural network and support vector machine. The results of experiments are analyzed in noise detection and filter scheme. It is shown that the MSN filter is superior to other techniques and results in a lower MSE and higher PSNR, IEF, and SSIM for a wide range of noise density. We
conclude that both noise detection and filter scheme are of great importance and impact the final quality of the restored image both visually and quantitatively. Although some images with low noise density do not give better over detection results using MSN, this case can be avoided by choosing better representative training examples. Therefore, MSN is a robust learning-based switching median filter for highly corrupted color images.
CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

Margin setting is a supervised learning algorithm that has salient performance, comparing to modern learning algorithms, such as support vector machine and neural networks. A comprehensive study of margin setting, including the algorithm, performance, as well as a new application is presented in this dissertation.

As for the algorithm, we mathematically present the algorithms in detailed steps and use a simple example to go through this algorithm, as a tutorial for margin setting.

Regarding the performance, a novel approach that combines margin concept in the support vector machine to spherical classification in margin setting is proposed. We choose margin, as a design parameter, to study the performance of margin setting and compare it with SVM. In particular, margin definitions are thoroughly presented. Moreover, we analyze the margin impact on training performance and generalization for all margin-based algorithms. Since margin setting is intrinsically a spherical classifier, margin impact are analyzed on general spherical classifiers first. We investigate one-sphere and multi-sphere performance as a fundamental research on margin setting as a
spherical classifier. Later, we theoretically analyze margin impact as discriminant functions using solution region, then extend our theoretical analysis to specific margin setting and SVM algorithm. To prove our theoretical findings, we conduct a set of experiments on both toy data sets and benchmark data sets.

Regarding application of the margin setting algorithm, a novel learning-based switching median filter is proposed to suppress impulse noise. The performance of our proposed filter outperforms SVM and Neural Networks, revealing that margin setting is a powerful machine learning algorithm.

7.2 Future Research

Margin setting is still a relative new algorithm that has not been adequately studied, so there are several promising directions that are related to the future work of this dissertation.

For margin setting algorithm, we can improve its performance by adding kernel mapping. Since all classification work is still processed in original feature space, it is nice to map the original data to higher feature space to deal with difficult situations. Besides, non-classification data should be addressed in the future by modifying the algorithm.

The convergence analysis of margin setting is necessary to mathematically measure its time complexity. In essence, margin setting is a stochastic evolutionary algorithm, and its computational complexity can be analyzed using some technics, such as Markov Chain [103].

Last, the era of “big data” brings the necessities of processing the large-scale data sets in machine learning applications. Learning algorithms should be scalable to handle the extremely massive data sets in the training phase. Machine learning algorithm, like
SVM, has been parallelized using MapReduce, etc [104-106]. Margin setting can be extended to “Cloud” margin setting to handle huge data sets in cloud computing environments.
% Yi Wang 2014
% Department of Electrical and Computer Engineering
% University of Alabama in Huntsville
% Margin Setting Main Program

function Prototype_Radius_vect =

margin_setting_trainner_multi_class(Na,N_Mutation_max,NGenMax,margin,varargin)

nClass = length(varargin);

if nClass < 2
    error(message('should have more than two classes to classify'));
else
fprintf('There are %d classes for pattern recognition.\n', nClass);

end

Class_set=cell(1,nClass);

% Unclass is the unclassified training points, i.e. missing points
Unclass = cell(1,nClass);

Dist = cell(1,nClass);

Class_set = varargin;

Class_set0 = varargin;

Total_set =[];

Prototype_Radius_vect = cell(1,nClass);

for k=1:nClass
    [~, Dim_C_S(k,:)] = size(Class_set{k});
    [T_Set_Size(k,:), Dim_C(k,:)] = size(Class_set{k});
    Total_set=cat(1,Total_set,Class_set{k});
end
% Dim\_C\_S is the number of attributes, each classes of training sets have
% the same number of attributes.
Dim\_C\_S = max(Dim\_C\_S);

% Initialize Generation Counter
NGen = 0;

N\_set = zeros(Na,Dim\_C\_S);

while (sum(T\_Set\_Size)>0)

% Storing prototype generated in one generation
Prototype\_Radius\_vect\_gen = cell(1,nClass);

% Generate of the set of random points N\_set
for u=1:Dim\_C\_S
    min\_value(1,u)=0;
    max\_value(1,u)=1;
    N\_set(:,u)= min\_value(1,u)+(max\_value(1,u)-min\_value(1,u)).*rand(Na,1);
end
% Initialize stopping flag for each class within one generation

    Stop_Mutation_Flag = zeros(nClass,1);

% Increment Generation Counter

    NGen = NGen + 1

% Initialize figure merit between two continuous mutations

    LFk0 = zeros(nClass,1);
    LFk1 = zeros(nClass,1);

% Initialize Mutation Counter for each class

    N_Mutation = zeros(nClass,1);

% Calculate distance from N_set to each class set

    Dist = euclidean_distance_multi_class(N_set,Class_set);

% Still calculate the distance from N_set to original class set (not reduced sets)

    Dist0 = euclidean_distance_multi_class(N_set,Class_set0);

% Find the closest elements

    ClosestRadiiAndLocat = closest_random_multi_class(Dist,Dist0);
% Get the number of closest elements (NCE) for each class

for k=1:nClass
    [NCE(k,:,:),~] = size(ClosestRadiiAndLocat{k});
end

% Calculate figure of merit of all classes when NCE > 0

if sum(NCE(:)) > 0
    % figure of merit and corresponding prototype radius
    [Fig_Merit0 LFk0 Prototype_Radius0] = figure_merit_multi_class(ClosestRadiiAndLocat, Dist,N_set);
end

% Enter Mutation

N_set1 = mutation_multi_class(ClosestRadiiAndLocat,N_set,Fig_Merit0,min_value,max_value);

mut = 0;

while (true)
    % Counter to count the maximum number of mutations of all classes
    mut = mut + 1
% Store the previous mutation's figure merit, and prototype_radius after
% the second mutation, i.e. count>1

if(mut >1)
    LFk0 = LFk1;
    Prototype_Radius0 = Prototype_Radius1;
end

% Increment Mutation Counter of the class (whose stopping mutation flag is
false 0)

for k = 1:nClass
    if(Stop_Mutation_Flag(k)==0)
        N_Mutation(k)= N_Mutation(k) +1;
    end
end

% check if the mutated N_set1 performs better than N_set

% Initialize Fig_Merit and Prototypes for the next mutation

Fig_Merit1 = cell(1,nClass);
Prototype_Radius1 = cell(1,nClass);

% Calculate Fig_Merit1, LFk1, and Prototype_Radius1

for k= 1:nClass
if(~isempty(N_set{k}))

Dist1 = euclidean_distance_multi_class(N_set{k},Class_set);

% Still calculate the distance from N_set to original class set (not reduced sets)
Dist10 = euclidean_distance_multi_class(N_set{k},Class_set0);

[Fig_Merit1{k} LFk1(k,:) Prototype_Radius1{k}] =
figure_merit_mutation_calculation(Dist1, Dist10, N_set{k}, k);

else

Fig_Merit1{k}=[];
LFk1(k,:)=0;
Prototype_Radius1{k}=[];
end
end

% hold on, plot_prototype(Prototype_Radius1,'k');

% 1) Stopping Mutation Condition:

% check if LFk1 > LFk0
% Update Stop_Mutation_Flag if needed

Pre_Stop_Mutation_Flag = Stop_Mutation_Flag;
Stop_Mutation_Flag = LFk1 <= LFk0;
for smi = 1: nClass

    if(Stop_Mutation_Flag(smi)==1 && Pre_Stop_Mutation_Flag(smi)==1)

        % in this case, you should stop mutation and all points of this
        % class has not been classified yet, go to the next generation

        Prototype_Radius_vect_gen{smi} = cat(1,

        Prototype_Radius_vect_gen{smi},Prototype_Radius0{smi});

        % update Stop_Mutation_Flag

    elseif(Stop_Mutation_Flag(smi)==0)

        % 2) Stopping Mutation Condition: Reaches max mutation index

        if(N_Mutation(smi)== N_Mutation_max)

            Stop_Mutation_Flag(smi)=1;

            Prototype_Radius_vect_gen{smi} = cat(1,

            Prototype_Radius_vect_gen{smi},Prototype_Radius1{smi});


        % 3) Stopping Mutation Condition: All/remaining class points are
        % classified (apply to first mutation)

        elseif(LFk1(smi) == T_Set_Size(smi))

            Stop_Mutation_Flag(smi) =1;

            Prototype_Radius_vect_gen{smi} = cat(1,

            Prototype_Radius_vect_gen{smi},Prototype_Radius1{smi});

            end

    end

end
for k = 1:nClass
    if (Stop_Mutation_Flag(k)==1)
        N_set1{k}=[];
    end
end

if (sum(Stop_Mutation_Flag)< nClass)
    % N_set = N_set1;
    N_set1 = mutation_again_multi_class(N_set1,Fig_Merit1,min_value,max_value);
else
    % [T_Set_Size Dim_C] = size(Class_set);
    break;
end
% Partition after mutation finishes, generate a new reduced training set

% enlarge the margin, shrink the radius

for k= 1:nClass
    if(~isempty(Prototype_Radius_vect_gen{k}))
        Rad_vect = Prototype_Radius_vect_gen{k}(:,Dim_C_S+1);
        Radius = (1 - margin)* Rad_vect;
        Prototype_Radius_vect_gen{k}(:,Dim_C_S+1)= Radius;
    end
end

Class_set1 = cell(1,nClass);

for k= 1:nClass
    [x ~] = size(Class_set{k});
    if(~isempty(Prototype_Radius_vect_gen{k}))
        Center = Prototype_Radius_vect_gen{k}(:,1:Dim_C_S);
    end
end
Radius = Prototype_Radius_vect_gen{k}((:,Dim_C_S+1);

for a = 1 : x
    R = norm_calc(Center',Class_set{k}(a,:))';
    if R >= Radius
        Class_set1{k} = cat(1,Class_set1{k},Class_set{k}(a,:));
    end
end
else % Prototype_Radius is empty, so this generation does not find any prototypes containing the reduced sets
    Class_set1{k} = Class_set{k};
end

[T_Set_Size(k,:), ~] = size(Class_set1{k});

end

Class_set = Class_set1;

% 1) Stopping Generation Condition: Reaches max generation index

T_Set_Size

if NGen == NGenMax
    Unclass = Class_set;
    break;
end
emp_proto = 1; % Indicate this generation is empty (1) or not empty (0).

% Empty generations does not generate any prototypes

for k = 1: nClass
    if(~isempty(Prototype_Radius_vect_gen{k}))
        emp_proto = 0;
    end
end

if(emp_proto == 0) % only catenate the non-empty prototypes
    Prototype_Radius_vect = cat(1,Prototype_Radius_vect,
    Prototype_Radius_vect_gen);
end

end

% Yi Wang 2014
% Department of Electrical and Computer Engineering
% University of Alabama in Huntsville

% This function randomly selects Na points and compute the distances
% between those points and each point of the training set
% The training set is a set of two arrays, one containing the class points
% and the second containing the non-class points
% Dist_c is the distance from the random points to the class and Dist_NC
% the array containing distances to the non-class array

function Dist = euclidean_distance_multi_class(N_set,Class_set)

[m nClass] = size(Class_set);
% [Tnc ~] = size(NonClass_set);
[Na ~] = size(N_set);

% create a cell storing the distances from random points to class points
Dist = cell(m,nClass);

if ( Na>0 )

for k=1:nClass

[mc, Dim] = size(Class_set{k});
if Dim == 0  % if this class is already classified, no training points exist
   Dist{k} = [];
elseif Dim == 1  % if one class only has one training point
   for in = 1 : Na
      PA = N_set(in);
   end
   %
end

end

end
for ic = 1 : mc
    Pnc = Class_set{k}(ic,:);
    Dist{k}(ic,in) = abs(PA - Pnc);
end

else

for in = 1 : Na
    PA = N_set(in,:);
    PA = reshape(PA,Dim,1);
    for ic = 1 : mc
        Pnc = Class_set{k}(ic,:);
        Pnc = reshape(Pnc,Dim,1);
        Dist{k}(ic,in) = norm_calc(PA,Pnc);
    end
end

end
%%this function compute the distance between N points to each class points
End

% Yi Wang 2014
% Department of Electrical and Computer Engineering
% University of Alabama in Huntsville

% This function is to measure the Euclidean distance of two arrays.
% PA and Pc are M*1 array
function [Distance] = norm_calc(PA,PC)

PA=double(PA);
PC=double(PC);
M = size(PA,1);

s=0;
for u = 1 : M
    s=s+(PA(u,1)-PC(u,1))*(PA(u,1)-PC(u,1)); % distance
end

Distance = sqrt(s);
function [ClosestRadiiAndLocat] = closest_random_multi_class(Dist, Dist0)

% [~, Bc] = size(Dist_C);
[m nClass] = size(Dist);

% Bc is the number of random points

% Closet_C is a M*3 array, and the first column, second column is the
% indices of the points(column, row) that choose from Dist_C, which has the
% smallest distance. column value of Closet_C is equal to the
% row number of the N_set, i.e. the location of sample points.

% locating the closest class points
for k=1:nClass

[~, Dim] = size(Dist{k});

if(Dim~=0)

for ic = 1 : Dim

    Closest_C{k}(ic,1) = ic;

    X1 = min(Dist{k}(:,ic));

    Closest_C{k}(ic,2) = find(Dist{k}(:,ic) == X1, 1, 'first');

end

end

end
Closest_C{k}(ic,3) = X1;

end

else  % this class has been classified.

Closest_C{k}=[];

end

end

% Get the distances for default class sets

for k=1:nClass

[~, Dim0] = size(Dist0{k});

for ic = 1 : Dim0

Closest_C0{k}(ic,1) = ic;

X1 = min(Dist0{k}(:,ic));

Closest_C0{k}(ic,2) = find(Dist0{k}(:,ic) == X1, 1,'first');

Closest_C0{k}(ic,3) = X1;

end

end

% set of random point that have a class point as their closest. Closest_R
% is an array containing the indices of the points that have a class point
% as their closest, the distance to the closest class point and the
% distance to the closest nonclass point

% T is the number of random points that have a class point as their closest

% The first column of ClosestRadiiAndLocat is the location of sample points
% that have a closest "Class" member

% The second column of ClosestRadiiAndLocat are the distances to the
% class

% The third column of ClosestRadiiAndLocat are the shortest distances to
% one of the non-class and it is the radius of ball of Class_sets

ClosestRadiiAndLocat = cell(m,nClass);
% ClosestRadiiAndLocatNC = [];

for ic = 1 : Dim
%
% if the distance of sample points to Class_set points is less than
% distance of sample points to NonClass_set points
% this point is the center of this Class,
% then radius is the closest distance from this point to non Class
% Y is to find centers
% Y is the distance from center to point, if
% this point belongs to class 2, this center is class 2 center

Y = [];  
Z = [];  

for k = 1: nClass
  if(~isempty(Closest_C{k})) % Find the first non-empty entry
    Y = Closest_C{k}(ic,3);  
in = k; % determine which points belong to which class's center
    break;
  end
end

% for k = 1: nClass
  if(~isempty(Closest_C{k}))
    if(Closest_C{k}(ic,3)< Y)
      Y = Closest_C{k}(ic,3);  
    % From each point, find its closest
    % distance to each class, record it.
    % e.g., if it is closest to
    % class2 points, it belongs to the
    % center of prototypes of class 2
end
in = k; % determine which points belonging to which class's center

end

end

end

% Z is to find radius

for k = 1: nClass
    if(~isempty(Closest_C0{k}) && k ~= in) % Find the first non-empty, initial as possible non-class
        Z = Closest_C0{k}(ic,3);
        jn = k; % determine which points belong to which class's center
        break;
    end
end

for k = 1: nClass
    if(k == in) % closest non-class
        if(~isempty(Closest_C0{k}));
            if(Closest_C0{k}(ic,3) <= Z)
                Z = Closest_C0{k}(ic,3); % From each point, find its closest
                % distance to each class, record it.
                % e.g., if it is closest to
% class 2 points, it belongs to the center of prototypes of class 2

jn = k; % determine which points belonging to which class's center

end

end

end

end

if(~isempty(Y) && (~isempty(Z)))
    C1 = [ic Y Z];
else
    C1 = [];
end

ClosestRadiiAndLocat{in} = cat(1, ClosestRadiiAndLocat{in}, C1);

end

% Yi Wang 2014
% Department of Electrical and Computer Engineering
% University of Alabama in Huntsville

% This program calculates figure of merit and corresponding prototype radius
Prototype_Radius in an M*4 array in which M is
% the number of prototypes generated, the first three columns represent
% the RGB coordinates and the 4th is the radius of the prototype

function [FigMerit LFk Prototype_Radius] =
figure_merit_multi_class(ClosestRadiiAndLocat, Dist,N_set)

 [~,nClass] = size(ClosestRadiiAndLocat);

% Create a cell storing the Figure merit values for each circle in each class,
% class 1 circles, class 2
% circles..class 3 circles... class n circles...

FM_values = cell(1, nClass);

for k=1:nClass

 [MCE(k,:,~)] = size(ClosestRadiiAndLocat{k});

 FM_values{k} = zeros(MCE(k,:),1);
 end

for k=1:nClass
if (~MCE(k,:) == 0)

for r = 1: MCE(k,:)
% We approximate it as Na as the maximum

    position = ClosestRadiiAndLocat{k}(r,1);
    radius = ClosestRadiiAndLocat{k}(r,3);

    FM_values(r,:) = cat(2,N_set(position,:),sum(Dist_C(:, position)<=radius));

    FM_values{k}(r,1) = sum(Dist{k}(:, position)<=radius);

    Prototype_Radius(r,:) = cat(2,N_set(position,:),radius);

end

else
% special case 1) if some class, there are no points are their centers, the
FM values are set to a 0.

    FM_values{k} = 0;

end

end

%LFk is matrix storing the largest Figure of merit among all points
% The 1st row is the largest Figure of merit of first class
% The 2nd row is the largest figure of merit of second class
% ..... 

LFk = zeros(nClass,1);

for k = 1:nClass

    LFk(k,:) = max(FM_values{k});

end
% Store the ball which has the value Fk=LFk(largest figure of merit)
% prototype_Radius is a matrix storing the initial prototype for each
% class,

% The 1st row is the prototypes of first class
% The 2nd row is the prototypes of second class
% ..... 

% Prototype_Radius = [];
Prototype_Radius = cell(1,nClass);
for k = 1:nClass
    if(sum(FM_values{k}>0)>0) % if FM_values are more than 0 zero, it does
not apply special case 1)
        which_row=find(FM_values{k}==LFk(k,:), 1, 'first');
        pos = ClosestRadiiAndLocat{k}(which_row,1);
        Prototype_Radius{k}=
        cat(2,N_set(pos,:),ClosestRadiiAndLocat{k}(which_row,3));
    end
end
Fig_Merit = FM_values;
function N_set1 = 
mutation_multi_class(ClosestRadiiAndLocat,N_set,Fig_Merit,min_value,max_value)

 [~,nClass] = size(ClosestRadiiAndLocat);

 % N_set1 is a cell that storing the mutated random sets for each class set
 N_set1 = cell(1, nClass);

 % NE is the number of elements in ClosestRadiiAndLocat each class
 for k=1:nClass
    [NE(k,:),~] = size(ClosestRadiiAndLocat{k});
    FM_values{k} = zeros(NE(k,:),1);
 end

 % [Loc ~] = size(ClosestRadiiAndLocat);
for k = 1: nClass

if(~NE(k,:)==0)

% initialize for each class

Fig_Merit_sum = sum(Fig_Merit{k}(:,1));

if(Fig_Merit_sum~=0)

fk_sum = 0.0;
fk = 0.0;
fk_values = [];
N_set_temp = [];

% start calculating mutation

for r = 1 : NE(k,:)
% Approximate it the maximum as Na

position = ClosestRadiiAndLocat{k}(r,1);

f_values = Fig_Merit{k}(r,1);

fk = f_values/Fig_Merit_sum;

fk_sum = fk_sum + fk;

fk_values(r,:) = cat(2,position,fk, fk_sum);
end

% the first column of fk_values is the position,
% the second column of fk_values is the fk values,
% the third column of fk_values if the sum above fk values

[Loa ~]= size(fk_values);

Y = rand(1);

% Y = 0.5; % for test

index = find(Y<=fk_values(:,3), 1, 'first');

Pm = fk_values(index,1); % Got the position of the coordinates

Center = N_set(Pm,:);

[size_n,dim]= size(N_set);

attr_mutate = Center;

for n = 1: size_n

% Mutate this center

% generates a random scalar that is either 0 or 1, with equal probability

warning('off','comm:obsolete:randint');

out = randint;

% out =0; % test algorithm correctness using non-randomness

if out==0
m_p = 1;
else
m_p = -1;
end

for p = 1:dim

if(attr_mutate(1,p) > (min_value(1,p) + max_value(1,p))/2)
pertube(1,p) = max_value(1,p) -attr_mutate(1,p); % maximum perturbation
else
pertube(1,p) = attr_mutate(1,p);
end

end

attr_mutate = attr_mutate+m_p*pertube*rand(1);
N_set_temp(n,:)=cat(1,attr_mutate);
end

N_set1{k}= N_set_temp;
end
% Yi Wang 2014
% Department of Electrical and Computer Engineering
% University of Alabama in Huntsville

function [Fig_Merit LFk Mutated_Prototypes] =
figure_merit_mutation_calculation(Dist, Dist0, N_set1, current_class_index)

[Dim, ~] = size(N_set1);  %Calculate the number of elements in mutated N_set

[m nClass] = size(Dist);

Center = [];

Mutated_N_Prototypes = [];
Fig_Merit = [];

LFk = [];
if(Dim >0)

for in = 1 : Dim

    Center = N_set1(in,:); % Each elements of N_set is the center of the class of N_set

    Z=[];
    m=1;
    for k = 1: nClass

        if(k~= current_class_index) % closet non-class
            Z(m,:) = min(Dist0{k}(:,in)); % find the closest distance to each non-class
            m=m+1;
        end
    end
end

Radius = min(Z); % Radius is the shortest distances to all non-class

Mutated_N_Prototypes(in,:) = cat(2,Center, Radius); % Mutated Prototypes

    Fig_Merit(in,:) = sum(Dist{current_class_index}(,in)<= Radius);
end
% Find the largest figure merit
LFk = max(Fig_Merit(:));

% Store the ball which has the value Fk=LFk(largest figure of merit)
which_row=find(Fig_Merit==LFk, 1, 'first');

Mutated_Prototypes = Mutated_N_Prototypes(which_row,:);
end

% Yi Wang 2014
% Department of Electrical and Computer Engineering
% University of Alabama in Huntsville

function N_set2 =
mutation_again_multi_class(N_set1, Fig_Merit1, min_value, max_value)

[~,nClass] = size(N_set1);

% N_set1 is a cell that storing the mutated random sets for each class set
N_set2 = cell(1, nClass);

% NE is the number of elements in ClosestRadiiAndLocat each class
for k=1:nClass
    [NE(k,:),~] = size(N_set1{k});
end

for k = 1: nClass
    if(~NE(k,:)==0)
        % initialize for each class
        Fig_Merit_sum = sum(Fig_Merit1{k}(:,1));
        if(Fig_Merit_sum > 0)
            fk_sum =0.0;
            fk = 0.0;
            fk_values =[];
            N_set_temp = [];
            % N_set1=[];
            % start calculating
            m =1;
            for r = 1 : NE(k,:)
                position = r;
                f_values = Fig_Merit1{k}(r,1);
fk = f_values/Fig_Merit_sum;

fk_sum = fk_sum + fk;

if(fk > 0) % eliminate the figure merit = 0
    fk_values(m,:)= cat(2,position,fk, fk_sum);
    m=m+1;
end
end

% the first column of fk_values is the position,
% the second column of fk_values is the fk values,
% the third column of fk_values if the sum above fk values

[Loa ~]= size(fk_values);

Y = rand(1);
% Y = 0.5; % for test

index = find(Y<=fk_values(:,3), 1, 'first');
Pm= fk_values(index,1); % Got the position of the coordinates
Center = N_set1{k}(Pm,:);

[size_n,dim]= size(N_set1{k});

attr_mutate = Center;
for n = 1: size_n

    % Mutate this center

    % generates a random scalar that is either 0 or 1, with equal probability
    warning('off', 'comm:obsolete:randint');
    out = randint;
    % out = 0; % test algorithm correctness using non-randomness
    if out == 0
        m_p = 1;
    else
        m_p = -1;
    end

    for p = 1:dim

        if(attr_mutate(1,p) > (min_value(1,p) + max_value(1,p))/2)
            pertube(1,p) = max_value(1,p) - attr_mutate(1,p); % maximum perturbation
        else
            pertube(1,p) = attr_mutate(1,p);
        end

end
end

attr_mutate = attr_mutate+m_p*pertube*rand(1);
N_set_temp(n,:)=cat(1,attr_mutate);

end

N_set2{k} = N_set_temp;
end
end
end
REFERENCES


[37] L. Reyzin and R. E. Schapire, "How Boosting the Margin Can Also Boost


References


Sons, 2012.


[86] C. Cortes, M. Mohri, and A. Rostamizadeh, "Generalization Bounds for Learning


