STUDIES ON SUPPORT VECTOR MACHINES AND APPLICATIONS TO VIDEO OBJECT EXTRACTION

DISSERTATION

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Yi Liu, B.S., M.S.

* * * * *

The Ohio State University

2006

Dissertation Committee:

Prof. Yuan F. Zheng, Adviser
Prof. Stanley C. Ahalt
Prof. David E. Orin

Approved by

Adviser
Graduate Program in Electrical and Computer Engineering
ABSTRACT

Pattern classification is a fundamental problem under study in machine learning. During the past decade, Support Vector Machine (SVM), a learning scheme for classification, has drawn tremendous attention due to its theoretical merit and practical success. However, limitations still exist when SVM meets real-world applications. The major thesis of this dissertation is to introduce new formulations that are derived to overcome the limitations of SVM and thus extend its horizon in practice. Furthermore, based on SVM and the extensions a novel approach toward video object (VO) extraction is presented to add another practical dimension to this powerful learning machine.

The first extension to be introduced is $\psi$-learning. It is motivated by the observation that the theory of SVM, which is well developed for separable cases, becomes less solid when extended to nonseparable cases. By replacing the hinge loss function in SVM with a designed $\psi$ function, $\psi$-learning fully takes into account the generalization errors in nonseparable cases and consequently improves the classification accuracy in such situations.

The second limitation of SVM is the requirement of Boolean (or hard) memberships. To address this problem, we reformulate SVM to be a new learning machine named Soft SVM, which allows samples to belong to different classes by different degrees and adjust the classification boundary from them accordingly.
Thirdly, this dissertation considers the generalization of SVM from binary classification, which is the scenario the classifier is originally designed for, to multi-class as well as single-class scenarios. For the multi-class case, we introduce both static and dynamic reliability measures into the framework of the traditional one-against-all multi-class scheme, and then based on these reliability measures we propose a new decision strategy for a better one-against-all method. One-class classification, on the other hand, is one special problem that raises the issue of describing the target class rather than discriminating between classes as in the binary and multi-class problems. In the context of SVM, we propose a new one-class classifier named minimum enclosing and maximum excluding machine (MEMEM), which offers capabilities for both pattern description and discrimination.

In practice, run time is always a critical factor, and the problem of slow training of SVM has been a bottleneck. In this dissertation, we tackle this efficiency issue in the area of feature selection. Two steps are taken. First, a new criterion is proposed to effectively filter out non-essential features before each training step begins. Secondly, we dynamically maintain a subset of training samples and use them rather than all the available samples for every necessary training. As a result, the total computational load is significantly reduced.

Lastly, a novel approach toward VO extraction is presented. Each VO is considered as a class, and VO extraction is realized by classifying every pixel to one of the available classes. It is significantly different from the traditional approaches yet overcomes many of their shortcomings. SVM, \( \psi \)-learning, and Soft SVM are employed as the classifier and experimental results demonstrate the great potential of machine learning in the area of VO extraction.
To my beloved parents and dear husband
ACKNOWLEDGMENTS

The first thank goes to my adviser, Dr. Yuan F. Zheng, for all the guidance he has given me throughout my tenure as a graduate student. When I found the topics that interested me, he was there to encourage me. When I had setbacks, he was there to boost my confidence. He was also there to have inspirational discussions with me when I was totally at a loss for directions. Without him, this dissertation would never have taken place. I also extend my gratitude to Dr. Xiaotong Shen. I have benefited a great deal from his expert knowledge in the area of machine learning and it has truly been a honor to collaborate with him.

Special thanks to Dr. Stanley Ahalt and Dr. David Orin for serving on my committee. Their technical feedbacks and constructive comments make this dissertation better than it could be.

It has been my pleasure to work with my colleges at The Ohio State University. Dr. Jianyu Dong, Dr. Eric Balster, Mr. Zhigang Gao, Mr. Yuanwei Lao, Mr. Gaotham Tamma, thanks for their continual help and kind company in my daily life. It has been a wonderful journey because of them.

Last but not least, I would like to thank my dear husband, Zhijun, and my parents, whose loving support has carried me through all the highs and lows.
VITA

October 19, 1974 .......................... Born - Changsha, China

1997 ........................................B.S. Electronic Engineering and Information Science

2000 ........................................M.S. Electronic Engineering and Information Science

PUBLICATIONS

Research Publications


**FIELDS OF STUDY**

Major Field: Computer Engineering

Studies in:

- Signal Processing
- Computer Science
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Dedication</td>
<td>iv</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>v</td>
</tr>
<tr>
<td>Vita</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xiv</td>
</tr>
<tr>
<td>Chapters:</td>
<td></td>
</tr>
<tr>
<td>1. BACKGROUND</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Purpose of Study</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Learning from Examples</td>
<td>2</td>
</tr>
<tr>
<td>1.2.1 The Framework</td>
<td>2</td>
</tr>
<tr>
<td>1.2.2 Induction Principles</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Support Vector Machine</td>
<td>6</td>
</tr>
<tr>
<td>1.3.1 Linear Separable Cases</td>
<td>7</td>
</tr>
<tr>
<td>1.3.2 Linear Nonseparable Cases</td>
<td>9</td>
</tr>
<tr>
<td>1.3.3 Nonlinear SVM</td>
<td>11</td>
</tr>
<tr>
<td>1.4 Video Object Extraction: A New Application of SVM</td>
<td>12</td>
</tr>
<tr>
<td>1.4.1 Conventional Approaches for VO Extraction</td>
<td>13</td>
</tr>
<tr>
<td>1.4.2 VO Extraction and Classification</td>
<td>14</td>
</tr>
<tr>
<td>1.5 Organization of the Dissertation</td>
<td>16</td>
</tr>
</tbody>
</table>
6. SVM AND FEATURE SELECTION ............................................. 65
   6.1 Introduction ......................................................... 65
   6.2 FS_SFS: Filtered and Supported Sequential Forward Search .... 69
      6.2.1 Problem Statement of Feature Selection ................. 69
      6.2.2 Review of FS_SFS ........................................... 71
      6.2.3 F_SFS: Filtered_SFS Using a New Criterion .......... 72
      6.2.4 S_SFS: Supported_SFS in the Context of SVM .......... 74
      6.2.5 FS_SFS: the Integration of F_SFS and S_SFS ......... 76
   6.3 Experimental Results ............................................. 77
      6.3.1 Results on Synthetic Data 1 ............................ 78
      6.3.2 Results on Synthetic Data 2 ............................ 82
      6.3.3 Results on Real-World Data ............................ 84
   6.4 Summary .......................................................... 89

7. CLASSIFICATION-BASED VIDEO OBJECT EXTRACTION USING
   SOFT SVM AND 𝜉-LEARNING ............................................. 91
   7.1 Introduction ........................................................ 91
   7.2 Approach Overview ................................................. 94
   7.3 Feature Extraction ................................................. 96
      7.3.1 Local Features ........................................... 97
      7.3.2 Neighboring Features .................................... 98
   7.4 Multi-Layer Learning and Classification ....................... 99
   7.5 Classification at the Block Level ............................... 103
   7.6 The Pyramid Boundary Refining Algorithm .................... 107
      7.6.1 Initialization Step ...................................... 107
      7.6.2 Iteration Step .......................................... 109
   7.7 Extension to Multiple VO Extraction ............................ 115
   7.8 Experimental Results ............................................. 117
      7.8.1 Single Object Extraction Using 𝜉-Learning ............. 117
      7.8.2 Single Object Extraction Using Soft SVM .............. 125
      7.8.3 Multiple Object Extraction Using Multi-Category 𝜉 .... 133
   7.9 Summary .......................................................... 140

8. CONCLUSIONS AND FUTURE WORK ..................................... 141

Appendices:
A. PROOF OF EQ. (5.9) AND (5.10) ........................................... 144

Bibliography ................................................................. 147
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Comparison of the center $a$ and radius $R$ obtained by SVDD and MEMEM when 25 training samples are used.</td>
</tr>
<tr>
<td>4.2</td>
<td>Comparison of the center $a$ and radius $R$ obtained by SVDD and MEMEM when 100 training samples are used.</td>
</tr>
<tr>
<td>4.3</td>
<td>Comparison of SVDD, MEMEM and SVM on the Biomed data set trained by 100 training samples. The optimal parameters are listed in the parenthesis.</td>
</tr>
<tr>
<td>5.1</td>
<td>Comparison of classification errors (Boldface indicates the best performance).</td>
</tr>
<tr>
<td>6.1</td>
<td>Comparison of classification accuracy and run time with different values of $K_n$.</td>
</tr>
<tr>
<td>6.2</td>
<td>Ten data sets from UCI</td>
</tr>
<tr>
<td>6.3</td>
<td>Comparison of the average classification accuracy and run time between FS_SFS and SFS over 20 trials. The range of the number of features selected are given in the parentheses as (min, max). For BCW, Glass and Post-operative Patients data set, the linear SVM is employed. For the rest, nonlinear SVM is utilized and the radial basis function is adopted as the kernel function.</td>
</tr>
<tr>
<td>6.4</td>
<td>Comparison of the classification accuracy between FS_SFS, DFPA [79], SVM BFE ([78] for linear and [80] for nonlinear problems) and filter method using class separability as the filtering criterion (Boldface indicates the best performance).</td>
</tr>
</tbody>
</table>
7.1 Average and standard deviation of run time per frame using $\psi$-learning.

7.2 Average classification errors produced by SVM, S_SVM, and FSVM, and the relative error reduction yielded by S_SVM with respect to SVM ($R_1$) and FSVM ($R_2$).

7.3 Comparison of average classification errors.
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A linearly separable case, and (b) a linearly nonseparable case.</td>
</tr>
<tr>
<td>1.2</td>
<td>The interpretation of margin in the linear separable case.</td>
</tr>
<tr>
<td>2.1</td>
<td>$F_{GE}$ function for GE, $\psi$ function for binary $\psi$-learning, and $F_{SVM}$ function for SVM.</td>
</tr>
<tr>
<td>3.1</td>
<td>An illustration of how SVM and S_SVM handle two samples with different $\lambda$’s. Suppose we are given two training samples, $x_1$ (the solid circle) and $x_2$ (the triangle). The class labels are $\tilde{y}_1 = 1$ and $\tilde{y}_2 = -1$ with $\lambda_1 = 1$ and $\lambda_2 = 0.5$ respectively. (a) SVM, which only considers the binary memberships, yields the hyperplane right in the middle between $x_1$ and $x_2$. (b) S_SVM, on the other hand, shifts the hyperplane to the sample with smaller certainty $\lambda$. (c) For the two-sample scenario, the location of the hyperplane in general can be solved analytically and given as shown.</td>
</tr>
<tr>
<td>3.2</td>
<td>An illustration of the change of the support vectors and the hyperplane when the soft memberships are assigned differently to the same training samples. The dashed line is the hyperplane yielded by SVM while the solid line by S_SVM. The circles and triangles that are solid are the supported vectors, and the numbers indicate the certainty measure $\lambda$.</td>
</tr>
<tr>
<td>3.3</td>
<td>Comparison between $F_{SVM}$ (SVM) and $F_{S,SVM}$ (Soft SVM).</td>
</tr>
</tbody>
</table>
3.4 An example of two-dimensional linear classification. 80 training samples are randomly generated from the uniform distribution over the unit disk \( \{ x_i = (x_{i1}, x_{i2}) \mid x_{i1}^2 + x_{i2}^2 \leq 1 \} \) with the conditional probability \( P(Y_i = 1|x_i) = \frac{1+x_{i1}^2}{2} \) and \( P(Y_i = -1|x_i) = \frac{1-x_{i1}^2}{2} \). The samples in class 1 and -1 are plotted as dots and crosses, respectively. By using Eq. (3.14), the soft membership is obtained as \( y_i = x_{i1} \). The hyperplanes yielded by S_SVM and SVM are represented by the magenta and cyan lines, respectively. Obviously, in both cases S_SVM produces the perfect hyperplane while SVM has much larger deviation.

4.1 Discrimination between two classes. (a) SVM. (b) MEMEM.

4.2 The effect of parameter \( \gamma \) on the resulted sphere of MEMEM. The center is depicted as a red star. The circles in blue, magenta and cyan represent the enclosing ball \( B(a, R_1) \), the decision boundary \( B(a, R) \) and the excluding ball \( B(a, R_2) \), respectively.

4.3 (a) The class distribution of the synthetic data. The true boundary is the circle centered at \((5, 5)\) with radius 3. (b) The true boundary (solid line in red), the boundary produced by SVDD (the dot-dashed line in blue), and the boundary produced by MEMEM (the dashed line in magenta).

5.1 A three-class example to illustrate the motivation of considering the classifier’s reliability at the classification stage. (a) Three classes and the true boundaries (solid lines). (b) The linear boundary (the dashed line) that separates class 3 and non-class 3. (c) The linear boundary (the dashed line) that separates class 1 and non-class 1.

5.2 Comparison of the performance by including the classes one by one. (a) Image segmentation data set using 1-degree polynomial as the kernel function. (b) Wine recognition data set using the linear kernel function. (c) Iris plant data set using the linear kernel function. (d) Letter recognition data set using RBF as the kernel function.

6.1 Comparison between (a) filter methods, (b) wrapper methods, (c) and the proposed methods for feature selection.
6.2 The active training set of the 2-D case, which are circled, maintained by S_SFS. (a) $v_1$, which is the support vectors obtained by considering only feature $x^1$. (b) $v_2$, which is the support vectors obtained by considering only feature $x^2$. (c) The support vectors obtained by training SVM on $V = v_1 \cup v_2$. (d) The support vectors obtained by using all the training samples. 79

6.3 The active training set of the 3-D case, which are circled, maintained by S_SFS. (a) The supported vectors obtained when $F = \{x^1\}$. (b) The supported vectors obtained when $F = \{x^1, x^2\}$. (c) The supported vectors obtained when $F = \{x^1, x^2, x^3\}$. (d) The support vectors obtained by using all the training samples. 80

6.4 The points of feature components $x_i$. (a) The ideal points. (b) The points obtained by using SFS. (c) The points obtained by using FS_SFS. 81

6.5 A comparison of feature selection methods on a synthetic linear problem. 83

6.6 The results of letter recognition (A and B). (a) The change of the value of the objective function vs. the number of retained features. (b) The comparison of classification errors between FS_SFS and SFS. 85

6.7 Comparison of classification errors between FS_SFS and SFS. (a) The Wisconsin Breast Cancer data set. (b) The Johns Hopkins University Ionosphere data set. (c) The Glass Identification data set. 85

6.8 The stability of FS_SFS on the BCW data set. (a) The selected features (squared) during trials. (b) Each bar shows how many times the corresponding feature are selected over 20 trials. 88

6.9 The change of $R_a$, which is defined as the ratio of the size of the subset (used for training) to the size of the total training set, during the feature selection process for six data sets. 88

7.1 An overview of the proposed approach for single VO extraction. (a) The training phase. (b) The tracking phase. 94

7.2 The similarity between object (human body) and the background in Silent sequence. (a) Some background areas (in circles and ellipses) are similar to the woman’s face. (b) Some background areas (in circles and ellipses) are similar to the woman’s hair. 96
7.3 Eight-connected neighboring blocks of block $B_0$. ............................. 97

7.4 Examples of a hyperplane tree. (a) A general two-level hyperplane tree. (b) The hyperplane tree associated with the data shown in Fig. 7.5. ................................. 99

7.5 Illustration of the multi-layer method. (a) 100 training samples $S_i$ with coordinates $(x_1^i, x_2^i) = (\rho_i \cos \theta_i, \rho_i \sin \theta_i)$ are randomly generated in the left hand side circle of the unit disk. The sample $S_i$ is labeled as “background” if $\theta_i \in \left[\frac{2\pi}{3}, \frac{4\pi}{3}\right]$. Otherwise, it is labeled as “object”. (b) A hyperplane $(w = [28.1, 32.2]^T, b = -4.3)$ is obtained by training the first layer. (c) One additional hyperplane $HP^2 (w = [29.2, -22.8]^T, b = -1.5)$ is obtained by training the second layer. (d) A different hyperplane $(w = [5.4, 0.27]^T, b = 0.9)$ is obtained when the first layer is trained using SVM. ......................................... 101

7.6 The results of Mom & Daughter sequence after different layers. (a) The original frame where mom and daughter are the object of interest. (b) The extracted VO after the first layer. (c) The extracted VO after the second layer. ................................................. 104

7.7 The results of Flower Garden sequences after different layers. (a) The original frame where the houses are the object of interest. (b) The extracted VO after the first layer. (c) The extracted VO after the second layer. (d) The extracted VO after the third layer. ......................... 104

7.8 The extracted VO of Mom sequence using block-level classification when block size is $9 \times 9$. (a) Frame 1. (b) Frame 118. (c) Frame 138. ..................................................... 105

7.9 Considering “+” and “o” as symmetric and also ignoring the orientation, the tiles only exhibit 3 different transition patterns. The points surrounded by □ are the centering pixels of the blocks. ...................... 108

7.10 The boundary zones (the blank areas) and boundary seeds (△) determined for different patterns. ...................................................... 109

7.11 The diagram of the iteration step of the pyramid boundary refining algorithm. .................................................. 110
7.12 An example of BZ updating $(N_0 = 3)$. (a) A transition area with the $BZ^0$ and classified boundary seeds. (b) The updated class map $CM^1$. (c) The updated boundary zone $BZ^1$ with boundary seeds (△) shown in the middle.

7.13 Three transition tiles in $CM^1 (N_0 = 3)$. (a) The transition tile of pattern #1. (b) The transition tile of pattern #2. (c) The transition tile of pattern #3.

7.14 The extracted VO of different boundary resolutions when $N_0 = 3$ (the block size is $9 \times 9$). (a) The training frame (frame #1). (b) The extracted VO after initial block-level classification. (c) The extracted VO after the first iteration of the pyramid boundary refining algorithm. (d) The extracted VO after the second iteration of the pyramid boundary refining algorithm. (e) The extracted VO after the third iteration (pixel-wise) of the pyramid boundary refining algorithm. (f) The extracted VO obtained by pixel-by-pixel classification.

7.15 An overview of the proposed approach for multiple VO extraction.

7.16 The results of Akiyo. (a) The extracted VO of frame 12. (b) The extracted VO of frame 134. (c) The extracted VO of frame 220.

7.17 The results of Mom. (a) The extracted VO of frame 1. (b) The extracted VO of frame 118. (c) The extracted VO of frame 138.

7.18 The results of Mom & Daughter. (a) The extracted VO of frame 2. (b) The extracted VO of frame 11. (c) The extracted VO of frame 16. (d) The extracted VO of frame 24. (e) The extracted VO of frame 43. (f) The extracted VO of frame 103.

7.19 The results of Silent. (a) The extracted VO of frame 33. (b) The extracted VO of frame 64. (c) The extracted VO of frame 123. (d) The extracted VO of frame 145. (e) The extracted VO of frame 171. (f) The extracted VO of frame 220.

7.20 The results of Flower Garden. (a) The extracted VO of frame 2. (b) The extracted VO of frame 12. (c) The extracted VO of frame 65. (d) The extracted VO of frame 86. (e) The extracted VO of frame 126. (f) The extracted VO of frame 149.
7.21 Segmentation error with respect to frame numbers. (a) Akiyo Sequence. (b) Mom & Daughter sequence. (c) Flower Garden Sequence.

7.22 Applying SVM and S_SVM to Akiyo on the first layer.

7.23 The extracted VO of Akiyo after the second layer classification. (a) and (b) are from frame 4, and (c) and (d) from frame 126.

7.24 Applying SVM and S_SVM to Mom and Daughter using RBF kernel.

7.25 Applying SVM and S_SVM to Sun Flower Garden.

7.26 The extracted VO of Akiyo by different machines on the first layer.

7.27 The comparison of the classification accuracy among SVM, FSVM and S_SVM. (a) Akiyo. (b) Mom and Daughter. (c) Sun Flower Garden.

7.28 The results of Students. (a) The original frames. (b) The extracted VO #1. (c) The extracted VO #2.

7.29 The results of Trevor. (a) The original frames. (b) The extracted VO #1. (c) The extracted VO #2. (d) The extracted VO #3.

7.30 The results of Sun Flower Garden. Columns (a) and (d) are original frames. Columns (b) and (e) are the extracted VO #1. Column (c) is the extracted VO #2.

7.31 Comparison of classification errors between multi-category $\psi$-learning, one-vs-all, one-vs-one and DAG. SVM is the underlying binary classifier employed by one-vs-all, one-vs-one and DAG.

7.32 Comparison of classification errors between multi-category $\psi$-learning, one-vs-all, one-vs-one and DAG. Binary $\psi$-learning is the underlying binary classifier employed by one-vs-all, one-vs-one and DAG.
CHAPTER 1

BACKGROUND

The process of learning in humans can be found everywhere in our daily life, and the design of algorithms that endow computers the ability to learn from examples falls into the area of machine learning. One major component of machine learning is statistical learning theory (SLT) [1] [2], the framework of which encompasses fundamental learning problems such as pattern classification. Introduced in 1960’s, SLT however remained purely theoretical until middle 1990’s when a classifier named Support Vector Machines (SVM) captured huge success in a wide range of applications. Since then, SLT has been out of the theoretical world and into the reality.

1.1 Purpose of Study

As a powerful classifier as SVM has been proven, limitations of SVM still exist when it meets real-world problems which motivates the studies presented in this dissertation. For instance, SVM is originally designed for separable two-class classification, but nonseparable multi-class is the more general case in reality. It is also difficult to apply SVM to the applications for which the data exhibit partial or unclear class memberships because in the framework of SVM each sample belongs to either one class or the other without ambiguity. Slow training is another critical issue of
SVM especially in the process of feature selection when multiple times of training are needed. So the major thesis of the dissertation is to derive a set of new formulations for SVM so as to overcome those limitations and thus extend the horizon of SVM in practice. Furthermore, a new application, video object (VO) extraction that can benefit from the birth of SVM, is presented to add another practical dimension to this powerful learning machine. Before presenting specific research topics, we first give a brief introduction of SLT, SVM and VO extraction.

1.2 Learning from Examples

1.2.1 The Framework

The goal of learning is to infer the unknown dependency between certain properties of objects and the measurements of them. The term “object” here is a very broad concept. It can range from cells to airplanes, from images to music, from apples to engines. One example of learning in practice is to predict the type of disease (the property) of a patient (the object) given some medical measures (the measurements). The measurements are assumed to be observable for all objects while the properties, on the contrary, in general are available only for a subset of objects known as examples.

In the framework of machine learning [1, 2, 3], the measurements are denoted as input variables $x$ in the input space $X$ and the properties are denoted as output variables $y$ in the output space $Y$. The dependency to be estimated takes the form of a function $f : X \to Y$. Let $Z = \{(x_i, y_i) \in X \times Y | i = 1, \ldots, N\}$ be the set of $N$ examples. It is also assumed that $(x_i, y_i) \in Z$ are drawn identically and independently from $P(X, Y) = P(Y|X)P(X)$, where the probability distributions $P(X)$ and $P(Y|X)$ are fixed but unknown.
To decide which of many possible functions $f$ best describes the dependency observed in the training samples, one measures the loss or discrepancy $L(y, f(x))$ between the true response $y$ given an input $x$ and the predicted response $f(x)$. Then the risk, which is the expected loss incurred from using a particular prediction function $f$, is defined as:

$$R(f) = \int L(y, f(x)) dP(x, y). \quad (1.1)$$

Now with the notations and definitions in place, the problem of learning from examples can be formulated as to find the function $f^*$ that minimizes the risk functional $R(f)$ when the training set $Z$ is given and the distribution $P(X,Y)$ is fixed but unknown.

### 1.2.2 Induction Principles

Classification, regression and density estimation are three main problems encompassed in the general formulation stated above. In this dissertation, we only consider the problem of classification, for which the output variables $y$ take on two values $y = \{-1, 1\}$ and the loss function becomes

$$L(y, f(x)) = \begin{cases} 0, & \text{if } y = \text{sign}(f(x)); \\ 1, & \text{if } y \neq \text{sign}(f(x)). \end{cases} \quad (1.2)$$

For this loss function, the functional 1.1 provides the probability of misclassification, which can be rewritten as

$$R(f) = P[Yf(X) < 0] = \frac{1}{2} E[1 - \text{sign}(Yf(X))], \quad (1.3)$$

and is also called generalization error (GE). Usually, a family function $\mathcal{F}$ that the candidate functions are to be selected from are defined. The simplest family is linear functions, which take the form $f(x) = w^T x + b$. 

3
Since the probability distribution $P(X,Y)$ is unknown, the expected risk $R(f)$ cannot be computed. Methods have been investigated to provide principles to find the function $f^*$ which minimizes the risk $R$ among all $f \in \mathcal{F}$, and these methods are called induction principles.

The most commonly used induction principle is the empirical risk minimization induction principle (ERM principle). Based on the training set $Z$, ERM calculates the empirical counterpart of $1.1$

$$R_{\text{emp}}(f) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)).$$

(1.4)

Then instead of minimizing $R(f)$, ERM seeks the function $f^N$ that minimizes the empirical risk $R_{\text{emp}}(f)$ and approximate $f^*$ with $f^N$. The motivation for doing so is that

$$\forall f, R_{\text{emp}}(f) \xrightarrow{N \to \infty} R(f).$$

(1.5)

according to the law of large number.

Although intuitively plausible, the minimization of the empirical risk has some downsides. First of all, the solution is not necessarily unique. For training set that is linear separable, for example, there are infinite number of hyperplanes that achieves zero $R_{\text{emp}}(f)$. Secondly, the problems of underfitting and overfitting likely occur because the complexity of the function $f$ is not controlled in ERM. Thirdly, to be useful in practice where only finite examples are available, the non-asymptotic analysis of the quality of ERM is necessary. To address these issues, the induction principle of structural risk minimization (SRM) has been suggested.

The key idea of SRM is to introduce a trade-off between the quality of the approximation and the complexity of the approximating function. The question needed
to be answered before we proceed is how one can measure and control the complexity of the function class. Generally speaking, it is difficult to obtain this quantity theoretically, and therefore the approximations have been adopted instead. One of the popular choices is Vapnik-Chervonenkis dimension or VC dimension [2]. In the case of classification where $\mathcal{F}$ is a set of sign functions, VC dimension is the maximum number $h$ of vectors $x_1, \cdots, x_h$ which can be separated in all $2^h$ possible ways using functions in the this set.

Recall that the quantity $R(f)$ cannot be computed because the distribution $P(X,Y)$ is unknown. However, with the help of VC dimension, the upper bound of $R(f)$ in probability is derived by Vapnik and Chervonenkis as the following theorem [2][3]:

**Theorem 1:** Let $h$ denote the VC dimension of the function class $\mathcal{F}$ and let $R_{\text{emp}}$ be defined by (1.4) using the 0/1-loss. For all $\delta > 0$ and $f \in \mathcal{F}$ the inequality bounding the risk

$$R(f) \leq R_{\text{emp}}(f) + \sqrt{\frac{h \left( \ln \frac{2N}{h} + 1 \right) - \ln(\delta/4)}{N}}$$

holds with probability of at least $1 - \delta$ for $N > h$ over the random draw of training samples.

As one can see, when $N/h$ is large, the second term on the right hand side of inequality (1.6) is small, which justifies ERM since a small value of $R_{\text{emp}}$ guarantees a small value of $R$. However, if $N/h$ is small, the discrepancy between $R_{\text{emp}}$ and $R$ becomes large and thus minimizing $R_{\text{emp}}$ alone is not good enough.

As a consequence of this observation, the induction principle of SRM is suggested. First of all, a nested structure in $\mathcal{F}$ is constructed as $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \cdots \subset \mathcal{F}_k \cdots$ where $\mathcal{F} = \cup_k \mathcal{F}_k$. Then SRM suggests that for a given training set $Z$ one should choose
the element of structure $F_i$ and the particular function from $F_i$ for which the upper bound stated in (1.6) is minimized.

One of the main results of the theory of SRM is stated in the following theorem.

**Theorem 2**: For any distribution function $P(X,Y)$, the SRM method provides convergence to the best possible solution with probability one.

In the next section, we will consider in detail a learning algorithm that implements SRM induction principle.

# 1.3 Support Vector Machine

Support Vector Machines (SVM) is the most popular technique for classification over the last decade. It is built upon the SRM induction principle and has found huge success in a variety of applications, ranging from object recognition [4, 5, 6], communications [7] [8], to image/video analysis [9, 10, 11].
1.3.1 Linear Separable Cases

First, let us assume that the training set \( Z \) is linear separable as shown in Fig. 1.1(a), and the function class takes the linear form as

\[
\mathcal{F} = \{ f \mid f(x) = w^T x + b, y_i f(x_i) \geq 0 \text{ for } (x_i, y_i) \in Z \},
\]

where \( w \) is of the same dimensionality as \( x \) and \( b \) is a scaler. In other words, the functions are a set of separating hyperlanes in the input space. Because \( Z \) is linear separable, by appropriately scaling the vector \( w \) and \( b \), any function \( f \in \mathcal{F} \) can be rewritten in the following canonical form

\[
\mathcal{F} = \{ f \mid f(x) = w^T x + b, y_i f(x_i) = 1 \}
\]

\[
= \{ f \mid f(x) = w^T x + b, y_i f(x_i) \geq 1 \}.
\]

Here the condition \((x_i, y_i) \in Z\) is dropped for the sake of simplicity. To apply SRM, a nested structure is then defined:

\[
\mathcal{F}_{\Lambda_k} = \{ f_k \mid f_k(x) = w_k^T x + b, y_i f_k(x_i) \geq 1, \|w_k\|^2 \leq \Lambda_k^2 \}.
\]

Evidently, \( \mathcal{F}_{\Lambda_i} \subseteq \mathcal{F}_{\Lambda_j} \) when \( \Lambda_i \leq \Lambda_j \). It has been shown that the VC dimension of the class \( \mathcal{F}_{\Lambda_k} \) is bounded by [1]

\[
h \leq \min(\Lambda_k^2 R^2 + 1, d + 1),
\]

where \( R \) is the radius of the smallest hypersphere enclosing the data, and \( d \) is the dimensionality of input vector \( X \). One valuable insight of this result is that VC dimension can be independent of \( d \) and can be controlled by changing \( \Lambda_k \), which plays a paramountly important role in the case known as the curse of dimensionality when \( d \gg N \).
Combining the two bounds, (1.6) and (1.10), one can easily see that
\[ \downarrow \|w\|^2 \rightarrow \downarrow \Lambda^2_k \rightarrow \downarrow h \rightarrow \downarrow R(f), \]
and therefore in order to obtain a small value of \( R(f) \) it is wise to find the function for which \( \|w\|^2 \) is minimized, which is how SVM is born. From the quantitative point of view, the quantity \( 2/\|w\| \) is a measure of the complexity of the classification problem. Geometrically, it is the lower bound of the minimum distance between two classes with respect to hyperplanes, and is named margin as show in Fig. 1.2. Mathematically, SVM finds the optimal linear function by solving the following optimization problem.

\[
\begin{align*}
\text{minimize} & : & L(w) = \frac{1}{2} \|w\|^2 \\
\text{subject to} & : & y_i (w^T x_i + b) \geq 1, \ i = 1, \ldots, N.
\end{align*}
\]

By introducing Lagrange multipliers \( \alpha_i \geq 0 \), the solution to this optimization problem, denoted as \( \tilde{w} \) and \( \tilde{b} \), can be obtained as the saddle point of the Lagrange
functional
\[ L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{N} \alpha_i \left( y_i (w^T x_i + b) - 1 \right), \] (1.13)
which is to be minimized with respect to \( w \), \( b \) and maximized with respect to \( \alpha_i \). At the saddle point, it follows that
\[ \bar{w} = \sum_{i=1}^{N} \bar{\alpha}_i x_i. \] (1.14)

Here \( \bar{\alpha}_i \) satisfies
\[
\text{maximize : } W(\alpha) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i^T x_j) + \sum_{i=1}^{N} \alpha_i, \\
\text{subject to : } \sum_{i=1}^{N} y_i \alpha_i = 0, \alpha_i \geq 0, \ i = 1, \ldots, N. \] (1.15)

As for \( \bar{b} \), it can be determined from the Kühn-Tucker conditions
\[ \bar{\alpha}_i \left( \bar{y}_i (\bar{w}^T x_i + \bar{b}) - 1 \right) = 0, \ i = 1, \ldots, N. \] (1.16)

As one can see, the shape of the objective function of the optimization problem expressed in 1.15 is quadratic, and therefore only one global optimum exists, which is a major numerical attraction of SVM formulation. Also note that \( \bar{w} \) is a linear combination of the input vectors and only those vectors for which \( y_i (w^T x_i + b) = 1 \) holds result in nonzero \( \alpha_i \). In other words, only a relatively small fraction of the samples \( x_i \) are needed to determine the location of the optimal hyperplane. These points are termed support vectors and this is how SVM got its name.

### 1.3.2 Linear Nonseparable Cases

For linearly nonseparable cases as shown in Fig. 1.1(b), there is no such a hyperplane that is able to classify every training point correctly. However the idea of
margin in SVM can be generalized by introducing the concept of *soft margin*. The new optimization problem thus becomes:

$$\begin{align*}
\text{minimize} & : \quad L(w, \xi_i) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i \\
\text{subject to} & : \quad y_i (w \cdot x_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, N, 
\end{align*}$$

(1.17)

where \( \xi_i \) are called slack variables which are related to the soft margin, and \( C \) is the tuning parameter used to balance the maximization of the margin and minimization of the training error.

Similar to what has been done to the optimization problem (1.12), the Lagrange multipliers are introduced and problem (1.17) is transformed to a quadratic programming problem

$$\begin{align*}
\text{maximize} & : \quad W(\alpha) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i^T x_j) + \sum_{i=1}^{N} \alpha_i, \\
\text{subject to} & : \quad \sum_{i=1}^{N} y_i \alpha_i = 0, \\
& : \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, N. 
\end{align*}$$

(1.18)

The optimal multipliers \( \bar{\alpha}_i \) then determines \( \bar{w} \) as

$$\bar{w} = \sum_{i=1}^{N} \bar{\alpha}_i x_i,$$

(1.19)

and \( \bar{b} \) as

$$\begin{align*}
\bar{\alpha}_i \left( \bar{y}_i (\bar{w}^T x_i + \bar{b}) - 1 \right) & = 0, \\
(C - \bar{\alpha}_i) \bar{\xi}_i & = 0.
\end{align*}$$

(1.20)

Once SVM is trained, the class label of a sample \( x \) is decided at the classification stage according to which side of the hyperplane it resides. For both separable and
nonseparable cases, the decision function takes the same form as
\[
\hat{y} = \text{sign}[\bar{w}^T x + \bar{b}] = \text{sign}\left[\sum_{i=1}^{N} \alpha_i y_i \bar{x}_i^T x + \bar{b}\right].
\] (1.21)

1.3.3 Nonlinear SVM

For the applications where linear SVM does not produce satisfactory performance, nonlinear SVM is suggested. The basic idea of nonlinear SVM is to map \(x\) by a nonlinearly mapping \(\Phi(x)\) to a much higher dimensional space in which the linear SVM is applied and the optimal hyperplane is found. Based on the observation that only the inner product of two vectors is needed to either solve the quadratic programming problems (1.15) & (1.18), or to obtain the classification result by (1.21), explicit definition of \(\Phi(x)\) is not necessary. Instead, one can just choose the nonlinear mapping implicitly by introducing the so-called kernel function \(K(x_i, x_j)\) which defines the operation of the inner product of vectors \(\Phi(x_i)\) and \(\Phi(x_j)\). As a result, to extend linear SVM to the nonlinear case, one only needs to replace the quantity \(x_i^T x_j\) with \(K(x_i, x_j)\) in (1.15) and (1.18).

Among a variety of kernel functions available, the radial basis and polynomial function are often chosen for many applications:

- **Radial Basis Function (RBF)**

  \[
  K(x_1, x_2) = \exp\left(-\frac{\|x_1 - x_2\|^2}{\sigma^2}\right),
  \]

  where \(\sigma\) is the parameter controlling the width of the kernel.

- **Polynomial Function**

  \[
  K(x_1, x_2) = \left(x_1 \cdot x_2 + 1\right)^d,
  \]

  where \(d\) is the degree of the polynomial.
Accordingly, the class label of an unseen point $x$ is given by

$$
\hat{y} = \text{sign}[w^T \Phi(x) + b] = \text{sign}\left[\sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b\right].
$$

(1.22)

1.4 Video Object Extraction: A New Application of SVM

The past decade has witnessed rapidly growing interest in content-based functionalities of video data such as video editing, content-based image retrieval, video indexing, video event analysis, and etc. To facilitate these functionalities, MPEG-4 introduces the concept of video objects (VOs) [12] [13] and brings up the problem of VO extraction. Also referred as the problem of object “segmentation and tracking” in many papers, VO extraction is faced with the following challenges:

1. VOs are defined as semantic entities. Consequently, VOs may be arbitrary collections of images regions and therefore they are usually heterogenous in spatial features;

2. VOs cannot be represented only by the centroid because they are often nonrigid and deformable;

3. VO extraction needs to do more than localization. The pixel-wise resolution is required for segmenting or extracting objects from video frames.

The classic approaches for object tracking or object detection, on the other hand, aim at localizing the centroid of objects and only provide bounding boxes as the results. For this reason, they cannot be applied directly to VO extraction and hence new techniques have been developed.
1.4.1 Conventional Approaches for VO Extraction

In the literature, the problem of VO extraction are approached either in an automatic [14, 15, 16, 17, 18, 19, 20, 21] or semi-automatic fashion [22, 23, 24, 25]. Generally the automatic approaches consists of two sequential steps: segmentation and tracking, and according to the primary criterion for segmentation they can be roughly categorized into two classes: spatial-based and temporal-based methods. The spatial-based segmentation method partitions each frame into homogeneous regions with respect to color or intensities. Then every region is tracked through time using the motion information. Typical partitioning algorithms include morphological watershed [15], K-means clustering [16], region growing [17], and the recursive shortest spanning tree [18]. A major advantage of the spatial-based segmentation approach is that it can yield relatively accurate object boundary. However the computational complexity is quite high and thus limits their usage in real-time applications since the segmentation has to be done on the whole image for every frame.

The temporal-based approaches [19, 20, 21], on the other hand, utilize the motion rather than spatial information to obtain the initial position and boundary of video objects. Because the objects of interest are usually moving, change detection is the major scheme for segmentation that can be done on the inter-frame or background-frame basis. Due to the image noise, objects’ boundaries are often irregular and require to be refined using the spatial information of the image. As the boundary fine-tuning procedure involves only the segmented moving region instead of the whole frame, higher efficiency is achieved. Unfortunately smooth motion, which is the essential assumption of the temporal-based method, may not always hold. For instance, when the frame loss occurs during the transmission of video or the object
exhibits abrupt variation of motion, the performance degrades. Another innovative approach is to fuse the intermediate results obtained by using different methods of segmentation [14].

Because of the semantic meaning a VO may carry, it may actually consists of arbitrary collections of image regions which may undergo non-coherent motions. For example, a person who is waving is not homogeneous as a video object in terms of color or motion. For this reason semiautomatic video segmentation, which defines the objects through users' supervision and tracks them in an unsupervised manner, has received a lot of attention [22, 23, 24, 25]. Characterized as modeling-and-searching, the objects of interest are initially extracted with user's assistance and then a model representing the object is created. A variety of models have been proposed including: active contour [24], 2D mesh [26, 27], binary model [28], deformable templates [29], corners and lines [30], 2D regions [31] and etc. Then in the subsequent frames the object is tracked and located where the best match of the model is found.

Although the aforementioned methods provide satisfactory results for extracting VOs of homogeneous motion characteristics, being robust in complex situations with abrupt motions or occlusions still remains a challenge.

1.4.2 VO Extraction and Classification

Accuracy and complexity are two critical issues for VO extraction, which have to be traded off in practice. The spatial-based approach can yield relatively accurate object boundary, but the computational load is quite high since the segmentation has to be done on the whole image for every frame. In comparison, the temporal-based approaches often produce irregular object boundaries which need to be refined
using the spatial information. As the boundary fine-tuning procedure involves only the segmented moving regions instead of the whole frame, higher efficiency can be achieved. For the modeling-and-searching type of methods, the key to efficiency is to effectively predict where the object might be to reduce the searching area. Mostly realized by motion estimation based on the assumption of smooth motion, the prediction is not reliable when occlusion or abrupt motion occurs.

In recent years, a new group of approaches has emerged to treat VO extraction directly as an object/background classification problem [32, 33, 34]. We call them classification-based approaches. Each VO is considered as a class, and VO extraction is achieved by classifying every pixel to one of the available classes. The classification-based methods have the potential to achieve both accuracy and low complexity. The methods are accurate because powerful classifiers can be trained for specific object/background separation. In the meantime, no motion assumptions are imposed which makes the classification-based methods more robust when VOs are of complicated motion characteristics. Low complexity, on the other hand, is achieved through evaluating the classification function at each pixel which involves only simple calculations, e.g., $w^T x + b$ for linear SVM.

Evidently, in the framework of classification-based approaches higher accuracy of classification produced by better classifiers directly leads to better performance of VO extraction, and therefore which classification technique to use is key to success. Based on SVM and its extensions, we propose a multi-layer approach for semiautomatic VO extraction. To make the method more effective and efficient, other important issues such as the feature representation of VO and the algorithm’s extensions to multiple
VO scenarios are also addressed. Chapter 7 will present the proposed approach in details.

1.5 Organization of the Dissertation

In this chapter, we explained the concept of learning and gave a brief review of SVM, a powerful learning machine of both theoretical merits and practical success. We also introduced the problem of VO extraction as a novel application that calls for state-of-the-art learning algorithms. For the rest of this dissertation, we will present several attempts that extend the horizon of SVM in practice and demonstrate their advantages in the application of VO extraction.

The rest of this dissertation is organized as follows. Chapter 2 through Chapter 5 will present four new learning algorithms, namely \( \psi \)-learning, Soft SVM, minimum enclosing and maximum excluding machine (MEMEM), and reliable one-vs-all SVM. The motivation of these new methods is to overcome the limitations of SVM in the applications where data are nonseparable, uncertain, severely under-sampled, and drawn from more than two classes, respectively.

In Chapter 6 the problem of feature selection in the context of SVM is addressed. We propose a new method named Filtered and Supported Sequential Forward Search (FS_SFS) to improve both the accuracy and efficiency of the feature selection process.

Chapter 7 presents a classification-based method for single as well as multiple VO extraction. Experimental results are provided to demonstrate the robustness of this approach. Meanwhile, different learning algorithms such as SVM and \( \psi \)-learning are employed as the classifier and their performances are compared.
Finally, the major contribution of this thesis is summarized in Chapter 8, and future research that can be based on this research work is discussed.
CHAPTER 2

ψ-LEARNING

The key idea of SVM is based on the concept of margin maximization conceived in separable cases. When extended to nonseparable cases, this foundation, however, becomes less solid. In this chapter, we introduce a variation of SVM called ψ-learning which retains the interpretation of large margin in separable cases but performs better than SVM in nonseparable cases.

2.1 Introduction

“Margin” is a very important concept for the development of SVM. Geometrically, it is the lower bound of the minimum distance between two classes which measures the separation between them. When extended to nonseparable case, the concept of margin becomes vague because in such situations the data are mixed and there is not clean separation between the classes to begin with. As pointed out in [35], in nonseparable cases the issue of generalization errors becomes more important but it is not fully taken into account in the formulation of SVM. Motivated by this observation, ψ-learning is developed.

The fundamental feature of ψ-learning is that it constructs the decision function $f$ by directly minimizing the generalization error. In the meantime, the concept of
margin maximization is still kept in the formulation. As a result, while retaining the same solid foundation as SVM in separable cases ψ-learning has the potential to deliver better performance in nonseparable cases. The advantage of ψ-learning over SVM has been demonstrated both theoretically and experimentally [35].

2.2 ψ-Learning: Robust Learning for Binary Classification

Before we proceed, let us recall the problem statement of binary classification. Every sample is represented by a $d$-dimensional input vectors denoted as $x \in X \subseteq R^d$ and the class index of the sample is denoted as $y \in Y = \{-1, 1\}$. A linear classifier in the input space is described as an equation $f(x) = \omega^T x + b$ where $\omega \in R^d$ and $b \in R^1$. The process to find the optimal $\omega$ and $b$ based on a given set $Z = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ is called training. The term “optimal” is defined with respect to certain criterion, which is the fundament of a classifier. Once trained, the classifier will classify an input vector $x$ according to the sign of $f(x)$ as

$$\hat{y} = \text{sign}(f(x)) = \begin{cases} 
1 & f(x) \geq 0 \\
-1 & f(x) < 0.
\end{cases}$$

The starting point of ψ-learning is a direct consideration of the ultimate goal of any classifier, which as shown in Section 1.2 is to minimize the probability of misclassification, or GE

$$GE = P[Yf(X) < 0] = \frac{1}{2} E[1 - \text{sign}(Yf(X))]. \quad (2.1)$$

The empirical generalization error or training error, is equal to

$$\frac{1}{2N} \sum_{i=1}^{N} \left( 1 - \text{sign}(y_i f(x_i)) \right), \quad (2.2)$$

where $N$ is the size of the training set. Minimizing Eq. (2.2) complies with ERM induction principle. However, as the SRM induction principle suggests (Theorem
1), small EGE does not guarantee small GE especially when $N$ is small. Following the margin-maximization idea in SVM, the following objective function is obtained

$$
\frac{1}{2}\|\omega\|^2 + C \sum_{i=1}^{N} \left( 1 - \text{sign}(y_i f(x_i)) \right),
$$

(2.3)

where $C$ is the tuning parameter used to balance the separating margin and the training error. A large value of $C$ emphasizes the importance of the training error. By defining $F_{GE}(\cdot) = 1 - \text{sign}(\cdot)$, the objective function becomes

$$
\frac{1}{2}\|\omega\|^2 + C \sum_{i=1}^{N} F_{GE}(y_i f(x_i)),
$$

(2.4)

However, the objective function represented by Eq. (2.3) has a numerical problem. If we scale $\omega$ and $b$ by a same positive factor $S$, the new function $f(x) = \frac{\omega}{S} \cdot x + \frac{b}{S}$ would yield the same classification result for the same $x$ and in turn give the same training error. Meanwhile $\frac{1}{2}\|\omega\|^2$ decreases by $\frac{1}{S^2}$. In this way $\omega$ and $b$ continue to decrease until both reach the machine precision, and as expected the final solution to Eq. (2.3) turns out to be a meaningless function $f(x) = 0$. To overcome this
drawback, the function $F_{GE}$ is replaced by a $\psi$ function which satisfies

$$
\psi(u) = \begin{cases} 
2(1-u) & \text{if } 0 \leq u \leq 1, \\
F_{GE} & \text{elsewhere}.
\end{cases}
$$

(2.5)

$F_{GE}$ and $\psi$ functions are depicted in Fig. 2.1(a) and Fig. 2.1(b), respectively. As one can see, $\psi$ function stays the same as $F_{GE}$ except in the interval $[0, 1]$ where $\psi$ introduces positive penalty in order to eliminate the scaling problem.

With the $\psi$ function, the objective function of $\psi$-learning is yielded

$$
\text{minimize: } \frac{1}{2}\|\omega\|^2 + C \sum_{i=1}^{N} \psi(y_i f(x_i)).
$$

(2.6)

### 2.3 Comparing $\psi$-Learning with SVM

To compare the difference between $\psi$-learning and SVM, let us revisit the objective function of SVM, which is expressed in Eq. (1.17) as

$$
\text{minimize : } \frac{1}{2}\|\omega\|^2 + C \sum_{i=1}^{N} \xi_i \\
\text{subject to : } y_i(\omega \cdot x_i + b) \geq 1 - \xi_i, i = 1, 2, ..., N.
$$

(2.7)

It is easy to see that the slack variable $\xi_i$ satisfies:

$$
\xi_i = \begin{cases} 
0, & 1 - y_i f(x_i) \leq 0, \\
1 - y_i f(x_i), & 1 - y_i f(x_i) > 0.
\end{cases}
$$

(2.8)

With Eq. (2.8), the objective function of SVM can be rewritten as

$$
\text{minimize : } \frac{1}{2}\|\omega\|^2 + C \sum_{i=1}^{N} F_{SVM}(y_i f(x_i)),
$$

(2.9)

where

$$
F_{SVM}(u) = \begin{cases} 
2(1-u) & \text{if } u \leq 1 \\
0 & \text{if } u > 1
\end{cases}
$$

(2.10)
is the so-called hinge loss function. The plot of $F_{\text{SVM}}(u)$ function is displayed in Fig. 2.1(c). Comparing (2.6) with (2.9), one can see that $\psi$-learning and SVM have similar objective functions but with difference in the second term.

In the linearly separable cases as shown in Fig. 1.1(a), the inequalities $y_if(x_i) \geq 1$ are forced to be true in both $\psi$-learning and SVM. In this regard, we have $F_{\text{SVM}}(y_if(x_i)) = \psi(y_if(x_i))$ and hence the two approaches are essentially the same. In the linearly nonseparable case as shown in Fig. 1.1(b), the second term of the two approaches behaves differently towards the wrongly classified samples. In SVM, the samples will affect the location of the hyperplane depending on their distances to the decision boundary. In other words, they force the estimated boundary to move towards them. The farther the distance of a sample, the stronger the moving force towards the sample, reflecting the fact that $F_{\text{SVM}}$ is just a convex upper envelope of $F_{\text{GE}}$ and linearly proportional to its variable in the positive side. In contrast to SVM, the $\psi$ function keeps constant as $F_{\text{GE}}$ when the variable is larger than 1, which forces $\psi$-learning to treat the wrongly classified samples in the same way regardless of how far they are. Consequently, the hyperplane is robust against those samples. This major difference leads to $\psi$-learning outperforming SVM in linearly nonseparable cases.

### 2.4 Multi-category $\psi$-Learning

SVM is originally designed for binary classification, and the so is the $\psi$-learning presented in the previous section. Instead of directly estimating the conditional probabilities, both SVM and $\psi$-learning focus on the decision boundary, which, however,
makes it difficult to generalize their applications from the binary to multi-class scenario. In this section, we introduce multi-category $\psi$-learning, which retains the desirable properties of its binary counterpart.

In the framework of multi-category $\psi$-learning, the class label is coded as $y \in \{1, 2, \ldots, M\}$, and the decision rule is

$$y = \arg\max_{i=1,\ldots,M} f_i(x),$$

where $M$ is the number of classes and $f_i$ is the decision function of class $i$ for $i = 1, \ldots, M$. In other words, the classifier assigns the sample $x$ to the class $i^*$ with the highest value of $f_i(x)$. Again for the linear classifier $f_i(x) = w_i^T x_i + b_i$.

As a characteristic of multi-class problems, multiple comparisons between classes need to be performed. To simplify the notations, an $(M-1)$-dimensional function vector $g(x, y)$ and a multivariate sign function $\text{sign}(u)$ where $u = (u_1, \ldots, u_{M-1})$ are defined as follows

$$g(x, y) = (f_y - f_1, \ldots, f_y - f_{y-1}, f_y - f_{y+1}, \ldots, f_y - f_M),$$

$$\text{sign}(u) = \begin{cases} 
1, & \text{if } u_{\text{min}} = \min(u_1, u_2, \ldots, u_{M-1}) \geq 0; \\
0, & \text{if } u_{\text{min}} < 0. 
\end{cases}$$

As mentioned before, the most prominent feature of $\psi$-learning is the direct consideration of GE. Defined as the probability of misclassification, GE yielded by an $M$-class classifier is

$$\text{GE} = E[Y \neq \arg\max_{i=1,\ldots,M} f_i(x)].$$

It can be shown that with the notations of $g(x, y)$ and $\text{sign}(u)$ GE can be rewritten as

$$\text{GE} = \frac{1}{2} E[1 - \text{sign}(g(x, y))].$$
As one can see, the $M$-category classification reduces to the binary case when $M = 2$. However, the formulation of binary $\psi$-learning is expressed in the coding system where the class label $y \in \{-1, 1\}$. To adapt it to the coding system described above, let us rewrite the objective function of binary $\psi$-learning [36] as

$$\begin{align*}
\text{minimize} & : \frac{1}{2} \sum_{j=1}^{2} \|w_j\|^2 + C \sum_{i=1}^{N} \psi(y_i(x_i) - f_{3-y_i}(x_i)), \\
\text{subject to} & : \sum_{j=1}^{2} f_j(x) = 0 \text{ for } \forall x.
\end{align*}$$

(2.14)

Here the sum-to-zero constraint is invoked to eliminate the redundancy in $(f_1, f_2)$, and it can be shown that $f(x) = f_1(x) - f_2(x) = 2(w_1^T + b_1)$ where $f(x) = w^T + b$ is the linear function yielded by Eq. 2.6.

In analogy to Eq. (2.14) which is for binary classification, the multi-category $\psi$-learning is formulated as

$$\begin{align*}
\text{minimize} & : \frac{1}{2} \sum_{j=1}^{M} \|w_j\|^2 + C \sum_{i=1}^{N} \psi_M(g(x_i, y_i)), \\
\text{subject to} & : \sum_{j=1}^{M} f_j(x_i) = \sum_{j=1}^{M} (w_i^T x_i + b_i) = 0.
\end{align*}$$

(2.15)

The $\psi_M$ function here is a multivariate version of $\psi$ with $(M - 1)$ arguments which is defined as

$$\psi(u) = \begin{cases} 
0, & \text{if } u_{\min} \geq 1; \\
2, & \text{if } u_{\min} < 0; \\
2(1 - u_{\min}), & \text{if } 0 \leq u_{\min} < 1 
\end{cases}$$

(2.16)

The multi-category $\psi$-learning preserves the desired properties of its binary counterpart. More specifically, for any $x$ satisfying $\text{sign}(g(x, y)) = -1$, $\psi_M$ assigns a constant penalty which is in the same spirit as GE. As a result, it is less sensitive to outliers and offers better learning ability. The cost for both binary and multi-category $\psi$-learning, however, is the computational advantage because $\psi$ is not a convex function as $F_{\text{SVM}}$. Fortunately, the selection of the $\psi$ function makes it decomposable to
be the difference of two convex functions, and as a result an advanced optimization strategy called difference convex (d.c.) decomposition can be utilized [36] [37].

2.5 Summary

This chapter introduces $\psi$-learning, a newly developed learning machine which constructs the decision function $f$ by directly minimizing the generalization error. As a result, it has good generalization ability and can outperform SVM in nonseparable cases. In Chapter 7, binary $\psi$-learning as well as multi-category $\psi$-learning will be employed to tackle the problem of single and multiple VO extraction.
CHAPTER 3

SOFT SVM

In the framework of SVM, each sample belongs to either one class or the other. This requirement, however, makes it difficult to apply SVM to applications where samples exhibit partial or unclear class memberships. To address this problem, this chapter reformulates SVM to be a new learning machine which is capable of dealing with binary (or hard) as well as real-valued (or soft) class memberships.

3.1 Introduction

Despite the superior performance of SVM in solving many classification problems, SVM is yet limited to crisp classification scenario where each sample falls into either one class or the other without ambiguity. However, there are situations where the collected samples exhibit partial or unclear membership as they may belong to different classes by different degrees. As a matter of fact, ambiguous membership is a typical problem for a large number of applications such as climatic prediction [38], soil classification [39], remote sensing [40], ecological modeling [41], etc., where soft classification can capture the fuzzy nature of the data better than hard classification. Unfortunately, SVM lacks this ability. To address this problem, Fuzzy SVM (FSVM) has been developed [42], which associates each labeled training sample with a fuzzy
membership $s_i > 0$ and employ $s_i$ to weigh the corresponding penalty term in the objective function. FSVM extends the horizon of SVM, but the information embedded in the fuzzy membership is missing when the corresponding sample is correctly classified because the penalty term is non-zero only when misclassification occurs\(^1\). In this chapter, we propose Soft SVM (S\(_S\)VM) which takes account of the real-valued memberships no matter whether the samples are classified correctly or not. When the samples are classified to the wrong side, the errors are penalized in proportion to the non-binary memberships such that the less certain the class labels are the less important the misclassification. When the samples are correctly classified, they are allowed to influence the boundary by pulling it close or pushing it away depending on the relative magnitude of their memberships. In either case, the samples can make different contribution to the learning of the decision boundary.

### 3.2 Soft Memberships

Consider \( N \) training samples \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \), where \( x_i \) is the input vector and \( y_i \) is the corresponding class label. For SVM \( y_i \in \{-1, 1\} \) while for S\(_S\)VM \( y_i \in [-1, 1] \) is a \textit{real-valued} variable called the \textit{soft membership}.

In some applications it is more typical and may be more convenient to employ the fuzzy membership

\[
(x_1, m_1^+, m_1^-), (x_2, m_2^+, m_2^-), \ldots, (x_N, m_N^+, m_N^-),
\]

where two quantities \( m_i^+ \) and \( m_i^- \) satisfying

\[
0 \leq m_i^+, m_i^- \leq 1 \quad \text{and} \quad m_i^+ + m_i^- = 1
\]

\(^1\)Here the term \textit{misclassification} refers to the error vectors \([5]\), which include the samples that are wrongly classified and those that are correctly classified but lie inside the margin strip.
describe the partial memberships of class 1 and -1, respectively. It is easy to see that
the quantity \((m_i^+ - 0.5)\) plays a similar role in the range \([-0.5, 0.5]\) as \(y_i\) in the range
\([-1, 1]\). Based on this observation, a linear one to one mapping between the fuzzy
and soft membership can be established as
\[
y_i = 2(m_i^+ - 0.5) = 2m_i^+ - 1 = m_i^+ - m_i^-,
(3.3)
\]
and therefore they can be interchangeably used.

\(y_i\) can be considered as a slider moving between -1 and 1. The more it slides
toward 1 (-1) the more degrees \(x_i\) exhibits to be a member of class 1 (-1), or the more
certain we are about the fact that \(x_i\) belongs to class 1 (-1). When the membership
stays in the middle (\(y_i = 0\) or \(m_i^+ = m_i^- = 0.5\)), we have absolutely no idea which class
the sample \(x_i\) comes from.

In order to fit into the notations of SVM, we further decompose \(y_i\) into two parame-
ters as \(y_i = \tilde{y}_i\lambda_i\), where \(\tilde{y}_i = \text{sign}(y_i)\) is the binary class label as in SVM and \(\lambda_i = |y_i|\),
named certainty measure, contains the additional membership information. Now, the
training set can be rewritten as \((x_1, \tilde{y}_1, \lambda_1), \ldots, (x_N, \tilde{y}_N, \lambda_N)\) with \(\tilde{y}_i \in \{-1, 1\}\) and
\(\lambda_i \in [0, 1]\). The above decomposition of \(y_i\) leads to only a minor variation to the
original optimization process of SVM which is shown below.

### 3.3 Formulation of Soft SVM

In this section, we derives the detailed formulation of S_SVM. Similar to what we
did when introducing SVM, we start with the simple case of linearly separable sets,
and then extend it to the linearly nonseparable case which is more general in reality.
3.3.1 Linearly Separable Cases

Let us first consider a very simple case. Assume we have only two training samples $x_1$ and $x_2$. Also assume that $x_1$ comes from class 1 with full membership ($y_1 = 1$) while $x_2$ from class -1 with less certainty, say, $y_2 = -0.5$. Under the assumption that the uncertainties of the class labels are caused by the overlapping of the data near the separating boundary, the optimal hyperplane is expected to move toward the point $x_2$ to reflect the unbalanced memberships rather than stand right in the middle between $x_1$ and $x_2$ as yielded by SVM. To do so, we relax the constraints in Eq. (1.12) as

$$\tilde{y}_i (w^T x_i + b) \geq \lambda_i,$$  \hspace{1cm} (3.4)

and as a result the optimal hyperplane of S-SVM becomes the solution to the following optimization problem:

$$\text{minimize : } L(w) = \frac{1}{2} ||w||^2$$

subject to : $$\tilde{y}_i (w^T x_i + b) \geq \lambda_i, \; i = 1, ..., N.$$ \hspace{1cm} (3.5)

As in SVM, the samples that achieve the equality of the constraints are called the support vectors.

To illustrate how this modification is going to affect the position of the optimal hyperplane, we provide an example with only two samples in Fig. 3.1. Evidently, the orientation of the hyperplane remains the same as that of SVM, which keeps the same maximal margin between the two classes. In the meantime, the hyperplane moves closer to the less certain sample which has smaller $\lambda$. Moreover, the analytical solution reveals that the hyperplane is shifted to the exact location where its distances to the support vectors (the two samples themselves in this case) are proportional to the certainty measures $\lambda_i$ (Fig. 3.1(c)).
Figure 3.1: An illustration of how SVM and S_SVM handle two samples with different $\lambda$’s. Suppose we are given two training samples, $x_1$ (the solid circle) and $x_2$ (the triangle). The class labels are $\tilde{y}_1 = 1$ and $\tilde{y}_2 = -1$ with $\lambda_1 = 1$ and $\lambda_2 = 0.5$ respectively. (a) SVM, which only considers the binary memberships, yields the hyperplane right in the middle between $x_1$ and $x_2$. (b) S_SVM, on the other hand, shifts the hyperplane to the sample with smaller certainty $\lambda$. (c) For the two-sample scenario, the location of the hyperplane in general can be solved analytically and given as shown.

Figure 3.2: An illustration of the change of the support vectors and the hyperplane when the soft memberships are assigned differently to the same training samples. The dashed line is the hyperplane yielded by SVM while the solid line by S_SVM. The circles and triangles that are solid are the supported vectors, and the numbers indicate the certainty measure $\lambda$. 
Another difference between SVM and S_SVM is in the selection of the support vectors. Unlike in SVM, the support vectors produced by S_SVM may be different combinations of the training samples. A simple example is given in Fig. 3.2. It is easy to see that for SVM all the three points are the support vectors. Yet when S_SVM is applied, the result is that they may not all be depending on the soft memberships associated. Actually, when we have more than two training samples, it is unpredictable which samples will be selected as the support vectors even for a two-dimensional linear separable case. Nevertheless, as observed through a large number of experiments, a good thumb of rule is that the samples with relatively higher certainties have better chances to be the support vectors and thus make greater contributions to the learning of the optimal boundary.

3.3.2 Linearly Nonseparable Cases

In analogy to what SVM does in the linearly nonseparable case, we introduce the non-negative variables $\xi_i$, which satisfy

$$\tilde{y}_i(w^T x_i + b) \geq \lambda_i - \xi_i$$

(3.6)

to penalize the objective function when the training samples $x_i$ are misclassified. However, as mentioned before, the statements “$x_i$ belongs to $\tilde{y}_i$” have different confidence levels measured by $\lambda_i$ and we should worry more about the misclassification of the samples with higher $\lambda_i$. Thus in our S_SVM the error term $\xi_i$ is further modified to $\lambda_i \xi_i$ to differentiate the penalty imposed to the error, which yields the following formulation:

$$\text{minimize} : \quad L(w, \xi) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \lambda_i \xi_i$$

subject to : \quad $\tilde{y}_i(w^T x_i + b) \geq \lambda_i - \xi_i.$

(3.7)
As pointed out in Section 2.3, the minimization problem of SVM is equivalent to
\[
\text{minimize : } \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^{N} F_{\text{SVM}}(y_i f(x_i)).
\]  
(3.8)

Similarly, the optimization function (3.7) is equivalent to:
\[
\text{minimize : } \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^{N} F_{S_{\text{SVM}}} (\lambda_i, \tilde{y}_i f(x_i)),
\]  
(3.9)

where \( F_{S_{\text{SVM}}} (u_1, u_2) = 0 \) if \( u_2 \geq u_1 \) and \( u_1(u_1 - u_2) \) if \( u_2 < u_1 \).

The plots of \( F_{\text{SVM}} \) and \( F_{S_{\text{SVM}}} \) are shown in Fig. 3.3(a) and Fig. 3.3(b), respectively. As one can see, all the training samples are treated equally when \( \lambda_i = 1 \) and \( S_{\text{SVM}} \) is reduced to SVM. Another extreme is \( \lambda_i = 0 \). In that case, the penalty term \( F_{S_{\text{SVM}}}(0, u_2) \) would always be zero no matter it is classified as 1 or -1. As a result the sample \( x_i \) is disregarded and would have no contribution to the learning of the decision boundary, which makes perfect sense since \( \lambda_i = 0 \) implies total uncertainty about which class \( x_i \) belongs to and therefore \( x_i \) is not different from any other uncollected samples at all.
Similar to SVM, the optimization problem of S_SVM can be transformed into the dual problem

\[
\begin{align*}
\text{maximize:} & \quad \sum_{i=1}^{N} \lambda_i \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \tilde{y}_i \tilde{y}_j x_i^T x_j \\
\text{subject to:} & \quad \sum_{i=1}^{N} \tilde{y}_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq \lambda_i C.
\end{align*}
\]

Again the optimal \( \bar{w} \) of Eq. (3.7) is the linear combination of \( x_i \) as \( \bar{w} = \sum_{i=1}^{N} \bar{\alpha}_i \tilde{y}_i x_i \), where \( \bar{\alpha}_i \) denotes the optimal point of Eq. (3.10). As for the optimal \( b \), it can be determined from the following Kuhn-Tucker conditions:

\[
\bar{\alpha}_i \left( \tilde{y}_i (\bar{w}^T x_i + b) - \lambda_i + \xi_i \right) = 0, \quad i = 1, \ldots, N,
\]

\[
(\lambda_i C - \bar{\alpha}_i) \tilde{\xi}_i = 0, \quad i = 1, \ldots, N.
\]

From the derivation above, one can see the dual problem of S_SVM is a quadratic problem similar to that of SVM. The computational load of the new approach thus stays the same.

For the applications where linear S_SVM is not suitable, nonlinear S_SVM is suggested. Similar to nonlinear SVM, nonlinear S_SVM maps the input vector \( x \) nonlinearly to a much higher dimensional space in which the optimal hyperplane is derived by applying the linear S_SVM described above.

### 3.4 Discussions about Soft SVM

#### 3.4.1 Generating Soft Memberships

One important issue associated with S_SVM is how to assign the soft memberships appropriately, which, generally speaking, is application dependent. Here we offer a choice from the probability point of view.

33
Considered as a random variable, the class label $Y_i$ of each training sample $x_i$ is either 1 or -1 with certain conditional probability $P(Y_i = 1|X_i = x_i)$ and $P(Y_i = -1|X_i = x_i)$. We propose to utilize the conditional probability to describe the partial memberships in Eq. (3.1), which defines

$$\begin{align*}
    m_i^+ &= P(Y_i = 1|X_i = x_i) \\
    m_i^- &= P(Y_i = -1|X_i = x_i).
\end{align*}$$

(3.13)

Using the relationship (3.3), we obtain the soft membership $y_i$ as

$$y_i = m_i^+ - m_i^- = P(Y_i = 1|X_i = x_i) - P(Y_i = -1|X_i = x_i).$$

(3.14)

The maximal certainty $\lambda_i = |y_i| = 1$ is produced when

$$P(Y_i = 1|X_i = x_i) = 1 \text{ or } P(Y_i = -1|X_i = x_i) = 1.$$  

(3.15)

On the contrary, when

$$P(Y_i = 1|X_i = x_i) = P(Y_i = -1|X_i = x_i) = 0.5,$$

(3.16)

we have $\lambda_i = |y_i| = 0$ indicating the maximal uncertainty about the class label.

It should be mentioned here that the conditional probability may be obtained only for training samples based on the knowledge specified by the application, and therefore are not generally available for all $x$ to be classified. One example will be shown later in Section 7.8.2.

3.4.2 Invariance Analysis

The superior performance of S_SVM on a synthetic data using the membership measure described above is shown in Fig. 3.4, but in reality it is difficult or even impossible to precisely know the absolute value of the memberships. The conditional
Figure 3.4: An example of two-dimensional linear classification. 80 training samples are randomly generated from the uniform distribution over the unit disk \( \{ x_i = (x_{i1}, x_{i2}) \mid x_{i1}^2 + x_{i2}^2 \leq 1 \} \) with the conditional probability \( P(Y_i = 1|x_i) = \frac{1 + x_{i1}}{2} \) and \( P(Y_i = -1|x_i) = \frac{1 - x_{i1}}{2} \). The samples in class 1 and -1 are plotted as dots and crosses, respectively. By using Eq. (3.14), the soft membership is obtained as \( y_i = x_{i1} \). The hyperplanes yielded by S_SVM and SVM are represented by the magenta and cyan lines, respectively. Obviously, in both cases S_SVM produces the perfect hyperplane while SVM has much larger deviation.

probabilities in Eq. (3.14), for instance, are usually unknown and have to be estimated. Fortunately, the proposed S_SVM possesses a nice property which makes it less sensitive to the imprecision of the membership assignment. As will be shown below, S_SVM is invariant to the scale of \( \lambda_i \). In other words, the same hyperplane is to be produced as long as the relative magnitude of the memberships preserves.

Suppose each \( \lambda_i \) is scaled by a positive number \( S \) and let \( \lambda_i' = S\lambda_i \). It can be shown that \( (\bar{w}', \bar{b}') \), which is solutions to the new optimization problem

\[
\min_{w'} : L(w', \xi_i) = \frac{1}{2} \|w'\|^2 + C \sum_{i=1}^{N} \lambda_i' \xi_i,
\]

35
subject to : \[ \tilde{y}_i(w^T x_i + b') \geq \lambda_i - \xi_i \] (3.17)
satisfies
\[
\begin{aligned}
\bar{w}' &= S\bar{w} \\
\bar{b}' &= S\bar{b},
\end{aligned}
\] (3.18)
where \((\bar{w}, \bar{b})\) is the solution to Eq. (3.7). Since we have
\[
\bar{w}'^T x + \bar{b}' = 0 \iff S\bar{w}^T x + S\bar{b} = 0 \iff \bar{w}^T x + \bar{b} = 0,
\] (3.19)
it comes to the conclusion that the two hyperplanes which are described by \(\bar{w}^T x + \bar{b} = 0\) and \(\bar{w}'^T x + \bar{b}' = 0\) essentially represent the same decision boundary. This property of scale-invariant makes S_SVM more applicable in solving real-world problems since in reality the relative value of the memberships usually can be obtained more easily and more accurately than the absolute values.

3.4.3 Underlying Assumption

As mentioned before, the non-binary membership is not an unusual phenomena for real-world problems. The motivation of both FSVM and S_SVM is the same: to treat different samples differently according to their different memberships. S_SVM is not always more applicable whenever we are given a set of training samples \((x_i, \tilde{y}_i, \lambda_i)\) with \(\tilde{y}_i \in \{-1, 1\}\) and \(0 \leq \lambda_i \leq 1\). Whether S_SVM should be employed or not depends on how the memberships are generated. One fundamental feature of S_SVM is that the information of \(\lambda_i\) is utilized in the way that the optimal hyperplane moves toward the samples with smaller \(\lambda_i\). The underlying assumption for doing so is that the soft memberships are resulted from the ambiguous nature of the class labeling caused by the overlapping of the data near the true boundary. Another good example is the application of face recognition. Given a number of pictures of two faces, one could have two different cases:
case 1: the two faces look alike, and

case 2: some pictures were taken more recently and others are older.

Between the above two cases, case 1 is a better target application of S_SVM because of the ambiguity between the two faces. In case 2 ambiguity is not an issue, but the idea instead is that more recent pictures are more useful and therefore should be given more weighting. In the latter case, $\lambda_i$ can be used as the regularization parameters to weight the error terms differently and accordingly which leads to the following

$$\begin{align*}
\text{minimize: } & \quad L(w, \xi_i) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \lambda_i \xi_i, \\
\text{subject to: } & \quad \tilde{y}_i(w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0.
\end{align*}$$

(3.20)

Eq. (3.20) as a matter of fact is the FSVM as proposed in [42] where $\lambda_i$ replaces the notation $s_i$. Unlike S_SVM, the solution to FSVM is not scale-invariant. The scaling of $\lambda_i$ by $S$ is equivalent to substituting the regularization parameter $C$ to $SC$, which changes the originally desired balance between the margin and the classification error, and potentially leads to a new hyperplane.

However, for the applications where the labeling of the collected data is ambiguous because of the similarity feature they exhibit, we expect S_SVM to deliver better performance, and VO extraction will be presented as such an application in Section 7.8.2 to demonstrate the advantage of S_SVM over SVM.

### 3.5 Summary

Despite its great success in a large number of applications, SVM is yet limited to crisp classification scenarios. In this chapter, we present Soft SVM, a reformulated version of SVM which allows training samples to belong to different classes by different degrees without increasing the computational cost.
CHAPTER 4

MINIMUM ENCLOSEING AND MAXIMUM EXCLUDING MACHINE

SVM has strong discriminating capability for class recognition. However, the classes are fixed and predefined, which forces any sample to be in either class 1 or -1 even when it comes from a totally new class. As a result, SVM lacks the ability to describe the classes and reject uncharacteristic patterns. One-class SVM, on the other hand, provides good representation for the classes but overlooks the discrimination issue between them. This chapter presents a new one-class classifier named minimum enclosing and maximum excluding machine (MEMEM), which offers capabilities for both pattern description and discrimination. The properties of MEMEM are analyzed and the performance comparisons using synthetic and real data are presented.

4.1 Introduction

This chapter addresses the problem of one-class classification (OCC), or more specifically the description problem of a target class in the presence of negative samples or outliers. A typical application is face recognition and verification. The system needs to identify one or multiple clients (recognition) and perform reliable rejection of impostors (verification) as well. Available are some photos of the persons of interest
and few, or most likely no, photos of potential impostors which evidently are far from being the representatives of the non-target class. Similar applications include speaker identification, fingerprint verification, target recognition in SAR imagery and etc.

Most often those tasks are handled as a conventional classification problem. Aiming at distinguishing one class from others, the classification systems assume that every sample comes from one of the classes which have been predefined during the training process. As a result, difficulties arise when rejection output is required or some classes are severely undersampled, which unfortunately is the case for the problem under study. Even Support Vector Machines (SVM), the proven powerful tool for discrimination and has been popularly chosen to achieve the classification task for a wide range of applications, is known as not able to reject uncharacteristic samples well and therefore suffers from false alarms [43].

A very simple scheme using output threshold has been extensively adopted to achieve the task of pattern verification. However, as pointed out in [44], simply using a threshold does not adequately solve the problem. For this reason, the problem of class description or so-called one-class classification has attracted the attention of many researchers [45, 46, 47, 48, 49, 50]. By modeling the support of the class where the data predominantly reside, OCC can recognize the new samples that resemble the training samples and detect uncharacteristic ones, or outliers, to avoid the ungrounded classification. Capable of working even when the training samples are solely from the target class, OCC is especially valuable when it is difficult or expensive to collect the samples from the non-target class. By far, the most well-known OCC studied in the context of SVM is support vector domain description (SVDD) proposed by Tax and Duinin [46]. By seeking the minimum hypersphere that encloses all the data in
the target class, SVDD finds the descriptive area that covers the data and excludes the superfluous space that results in false alarms. Two alternative approaches are developed in [47] and [48] which use hyperplanes to describe the class. Under the assumption that the uncharacteristic samples lie on the same side of the hyperplane as the origin, [47] places the hyperplane such that it separates all the samples from the origin with the maximal distance. [48], on the other hand, tries to attract the hyperplane toward the center of the training samples. One common limitation of the two latter methods is the restriction for the samples to be unit norm.

In many recognition and verification systems, a two-step procedure is designed. First, the decision of acceptance or rejection is made by an OCC algorithm. Then the accepted sample will be further classified by conventional classifiers such as SVM into one of the predefined classes. The difficulty, however, is that the samples drawn from different classes may show great diversities and therefore do not necessarily cluster well. As a result, describing them as a whole by a single OCC is not able to deliver desired performance. A one-step approach is proposed in [51], which constructs $M$ one-class SVMs with one for each class. If all classifiers reject the input sample, the system rejects it; otherwise, it is assigned to the class with the highest confidence level of acceptance. Unlike the two-step procedure, each one-class classifier is now responsible for both recognition and rejection. However, because of the descriptive nature of OCC, the factor of the separation between classes is not considered in the formulation and therefore the discrimination issue is overlooked.

In this chapter, we propose a novel one-class classifier in the context of SVM, which is named the minimum enclosing and maximum excluding machine (MEMEM). Similar to SVDD, MEMEM models the support of the target class by a hypersphere, but
unlike SVDD it seeks an additional hypersphere that excludes the negative samples by a wide shell. By doing so, the discriminating ability of the classifier is enhanced while its descriptive ability is preserved.

4.2 MEMEM

For the problem of class description, an assumption is usually made that the separation boundary drawn by the classifier is closed [44]. SVDD, for example, imposes a hypersphere $B(a, R)$, which is characterized by the center $a$ and radius $R$, as the separating surface around the samples. The spherical shape of the boundary, however, is a rigid model which fortunately can be made flexible by using kernel functions as by SVM. For its simplicity, we also models the boundary as a hypersphere.

Suppose we are given $N$ training samples $(x_i, y_i)$ with $x_i \in \mathbb{R}^d$ and $y_i \in \{1, -1\}$, where the target class is defined as class 1. When the training samples are spherically separable, two concentric hyperspheres can be found such that all the samples from class 1 are enclosed by the inner hypersphere $B(a, R_1)$ and all the samples from class -1 are excluded by the outer hypersphere $B(a, R_2)$ as

\[
||a - x_i||^2 \leq R_1^2 \quad \text{for} \quad y_i = 1, \\
||a - x_i||^2 \geq R_2^2 \quad \text{for} \quad y_i = -1.
\] (4.1)

Similar to SVDD, we want the inner hypersphere $B(a, R_1)$ to be as small as possible for good description of the target class. In the meantime, to separate the two classes we want the outer hypersphere $B(a, R_2)$, which pushes the negative samples away, to be as large as possible, and we call the one with the largest radius the maximum excluding sphere. Recall in SVM, the discrimination between two classes is achieved by maximizing the margin as shown in Fig. 4.1(a). Inspired by this idea, to obtain
Figure 4.1: Discrimination between two classes. (a) SVM. (b) MEMEM.

good discriminating ability for the hyperspheres we maximize the width of the shell between $B(a, R_1)$ and $B(a, R_2)$ (the shaded area in Fig. 4.1(b)), which is proportional to the quantity $R_2^2 - R_1^2 \triangleq 2\Delta R^2$. Then the resulted boundary would be $B(a, R)$ where $R = \sqrt{(R_1^2 + R_2^2)/2}$ as shown as the dashed line in Fig. 4.1(b). In order to achieve the objective of both description (small $R^2$) and discrimination (large $\Delta R^2$), the simplest thing to do is to minimize the quantity $R^2 - \Delta R^2$.

With the motivations explained above, we now propose MEMEM as

$$\min_{a, R^2, \Delta R^2} : \gamma R^2 - \Delta R^2$$
subject to : $$||a - x_i||^2 \leq R^2 - \Delta R^2$$ for $y_i = 1$,

$$||a - x_i||^2 \geq R^2 + \Delta R^2$$ for $y_i = -1$$$(4.2)$$

where a non-negative parameter $\gamma$ is introduced to give users the flexibility to balance the importance of a small enclosing ball and a large shell. When the training samples are scarce, $\gamma$ should be set close to 1. One extreme is $y_i = 1$ for $i = 1, \ldots, N$, 42
which means no negative samples are available. In this case, $\gamma$ should be equal to 1 and MEMEM only cares about the minimization of the enclosing hypersphere, which reduces it to SVDD. On the other hand, if the classes are well sampled such that the decision boundary can be supported from both sides, $\gamma$ should be close to 0. By doing so, MEMEM lays emphasis on the separation of the two classes as conventional SVM. Fig. 4.2 shows how the resulted hypersphere changes when different $\gamma$ are used.

By introducing multipliers $\alpha_i$ for the inequality constraints, we obtain the duel problem of Eq. (4.2) as

$$
\max_{\alpha_i} : -\frac{1}{\gamma} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{N} \alpha_i y_i x_i^T x_i \\
\text{subject to : } \sum_{i=1}^{N} \alpha_i y_i = \gamma, \quad \sum_{i=1}^{N} \alpha_i = 1, \\
\alpha_i \geq 0, \text{ for } i = 1, \ldots, N.
$$

(4.3)

It can be shown that the center $a$ of the hypersphere is a linear combination of the training samples as $a = \sum_{i=1}^{N} \alpha_i y_i x_i / \gamma$. Similar to SVM, the optimization of (4.3) is a quadratic programming problem, and as a result many $\alpha_i$ are zeros. In other words, the center are determined by a few training samples and we also call them support vectors. It is easy to check that the parameter $\gamma$ should satisfy $0 < \gamma \leq 1$ for (4.3) to have feasible solutions.

When the samples cannot be separated by a sphere, we have to allow some negative samples inside the enclosing hypersphere and/or some positive samples outside the excluding hypersphere. In analogy to what SVM does in the situation, slack variables $\xi_i \geq 0$ are introduced and Eq. (4.2) becomes

$$
\min_{a, R^2, \Delta R^2, \xi_i} : \gamma R^2 - \Delta R^2 + C \sum_{i=1}^{N} \xi_i
$$
Figure 4.2: The effect of parameter $\gamma$ on the resulted sphere of MEMEM. The center is depicted as a red star. The circles in blue, magenta and cyan represent the enclosing ball $B(a, R_1)$, the decision boundary $B(a, R)$ and the excluding ball $B(a, R_2)$, respectively.
subject to : \[ ||a - x_i||^2 \leq R^2 - \Delta R^2 + \xi_i \text{ for } y_i = 1, \]
\[ ||a - x_i||^2 \geq R^2 + \Delta R^2 - \xi_i \text{ for } y_i = -1, \]
\[ \Delta R^2 \geq 0. \] (4.4)

Note one additional constraint \( \Delta R^2 \geq 0 \) is added in Eq. (4.4) to force the enclosing ball to be inside the excluding ball, which is not assured in the non-separable case.

The dual problem of (4.4) is very similar to that of (4.3). The differences lie in the the upper bound of \( \alpha_i \) and the additional multiplier \( \beta \) associated with the constraint \( \Delta R^2 \geq 0 \):

\[
\begin{aligned}
\max_{\alpha, \beta} & : -\frac{1}{\gamma} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{N} \alpha_i y_i x_i^T x_i \\
\text{subject to} & : \sum_{i=1}^{N} \alpha_i = \gamma, \sum_{i=1}^{N} \alpha_i - \beta = 1, \\
& \beta \geq 0, \ C \geq \alpha_i \geq 0, \text{ for } i = 1, \ldots, N. 
\end{aligned}
\] (4.5)

It can be shown that by choosing \( \gamma = 1 \), Eq. (4.4) becomes the soft-margin SVDD, and therefore with or without the negative samples, MEMEM has SVDD as one special case when MEMEM only deals with the task of description.

4.3 Experimental Results

4.3.1 Synthetic Data

Before conducting experiments on real-life data, we try MEMEM on a 2-D synthetic data set to illustrate the performance. The data are uniformly distributed in a square as shown in Fig. 4.3(a). The triangles and the dots represent the target class and the non-target class, which are the samples inside and outside the ball centered at (5, 5) with radius 3, respectively. First, we randomly generate 26 training samples
Figure 4.3: (a) The class distribution of the synthetic data. The true boundary is the circle centered at (5, 5) with radius 3. (b) The true boundary (solid line in red), the boundary produced by SVDD (the dot-dashed line in blue), and the boundary produced by MEMEM (the dashed line in magenta).

with 13 from each class. The samples generated in one simulation are depicted in Fig. 4.3(b), where one can see that the training samples are quite sparse. The decision boundaries produced by SVDD and MEMEM ($\gamma = 0.5$), which are averaged over 100 simulations, are listed as the second row in Table 4.1 and drawn along with the true boundary in Fig. 4.3(b). Evidently, MEMEM outperforms SVDD.

Unbalanced training data are also generated. We test two cases: (1) $\eta = 5/2$; and (2) $\eta = 2/5$, where $\eta$ denotes the ratio of the number of training samples in the target class and the non-target class. The average center and radius of the ball produced by SVDD and MEMEM are listed in Table 4.1. Both SVDD and MEMEM find the center very well, which we believe is because the shape of the true boundary (a circle) fits well the model chosen by both methods. The major difference, however,
Table 4.1: Comparison of the center $a$ and radius $R$ obtained by SVDD and MEMEM when 25 training samples are used.

<table>
<thead>
<tr>
<th>$\eta$</th>
<th>SVDD</th>
<th>MEMEM</th>
<th>$\eta$</th>
<th>SVDD</th>
<th>MEMEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1:1$</td>
<td>$(5.06, 5.01)$</td>
<td>$(5.05, 4.98)$</td>
<td>$1:1$</td>
<td>$(5.06, 5.01)$</td>
<td>$(5.05, 4.98)$</td>
</tr>
<tr>
<td>$2:5$</td>
<td>$(5.04, 4.99)$</td>
<td>$(5.05, 4.97)$</td>
<td>$2:5$</td>
<td>$(5.04, 4.99)$</td>
<td>$(5.05, 4.97)$</td>
</tr>
</tbody>
</table>

Table 4.2: Comparison of the center $a$ and radius $R$ obtained by SVDD and MEMEM when 100 training samples are used.

is the radius. Due to its descriptive nature, the emphasis of SVDD is to make the ball as small as possible and therefore it produces a much smaller ball than the true one. MEMEM, on the other hand, also considers the separation between the classes. As a result, the resulted boundary is very close to the true boundary even when the samples are scarce.

To test how MEMEM performs when the training set is relatively abundant, we conduct the second series of simulations by utilizing 100 training samples. Again, we run both balanced and unbalanced data, and list the results in Table 4.2. As one can see, with more training data, the ball found by SVDD is expanded and closer to the true boundary. However, it is still outperformed by MEMEM, which almost produces the perfect decision boundary.

47
4.3.2 Medical Diagnosis Data

For the real-life data, we investigate the performance of MEMEM on the Biomed data set [52] from the Statlib data archive [53], which was collected in a study that aims at developing screening methods to identify carriers of a rare genetic disorder. Excluding 15 observations that are not usable due to missing values, this biomedical data set contains 194 observations with 127 normal samples from healthy patients and 67 abnormal samples from disease carriers. Each observation has 4 attributes corresponding to measurements taken on blood samples.

First, we train MEMEM with the balanced data ($\eta = 1$). In total, 100 samples are randomly chosen with 50 each from normal and abnormal classes. We are left with 77 normal observations and 17 abnormal observations, which are used as the test samples. RBF, which is reported as a good choice for this data set [48], is adopted as the kernel function. The generalization accuracy is estimated by using different parameters in Eq. (4.4). More specifically, we try $C = [0.1, 0.2, \ldots, 1]$ and $\gamma = [0.05, 0.1, 0.2, \ldots, 1]$, which provides $10 \times 11 = 110$ combinations, to find the best pair $(C, \gamma)$ that yields the highest classification accuracy on the test samples.

SVDD and SVM are also employed to make the performance comparison. Different from MEMEM, SVM and SVDD have only one parameter to tune, which shares the same notation as $C$ in the formulation. For SVM, we test 13 choices as $C = 10^{-1} \cdot [2^0, 2^1, \ldots, 2^{12}]$. SVDD, however, requires the parameter to be $0 \leq C \leq 1$ and thus we vary $C$ from 0 to 1 with a step size of 0.1. Again, each machine is trained for every choice of $C$ and the highest classification accuracy is selected for the performance comparison.
<table>
<thead>
<tr>
<th>$\eta$</th>
<th># of training samples</th>
<th>SVDD $(C)$</th>
<th>MEMEM $(C, \gamma)$</th>
<th>SVM $(C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>normal</td>
<td>abnormal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 : 1</td>
<td>89</td>
<td>11</td>
<td>76.3% (0.6)</td>
<td>81.4% (0.1, 0.7)</td>
</tr>
<tr>
<td>4 : 1</td>
<td>80</td>
<td>20</td>
<td>81.4% (0.8)</td>
<td>88.7% (0.8, 0.3)</td>
</tr>
<tr>
<td>2 : 1</td>
<td>67</td>
<td>33</td>
<td>79.4% (0.1)</td>
<td>89.7% (0.9, 0.3)</td>
</tr>
<tr>
<td>1 : 1</td>
<td>50</td>
<td>50</td>
<td>82.5% (0.6)</td>
<td>92.8% (0.05, 0.1)</td>
</tr>
</tbody>
</table>

Table 4.3: Comparison of SVDD, MEMEM and SVM on the Biomed data set trained by 100 training samples. The optimal parameters are listed in the parenthesis.

Other scenarios that test how MEMEM performs on unbalanced data set are also simulated. The description of the data shows that because the disease is rare, there are only a few carriers of the disease from whom data are available. For this reason, we always have the training set contain more normal observations than abnormal ones to make the simulations practical. Three cases are experimented and the results are listed in Table 4.3. As one can see, for all the scenarios simulated, MEMEM consistently performs the best or comparably the best among the three learning machines. When $\eta = 1$, SVM and MEMEM yield comparable performance. This is because with ample normal and abnormal samples, this problem can be approached by a traditional binary classifier. However, when $\eta = 8 : 1$ for which the number of abnormal samples are relatively few, SVM, which focuses on discrimination between classes, is outperformed by both SVDD and MEMEM. Actually, the accuracy yielded by MEMEM, which is 81.4%, is significantly higher than that of the other two machines, which are 76.3% and 72.3% respectively.
4.4 Summary

MEMEM, a new algorithm for robust classification, is proposed. Rooted in one-class SVM, MEMEM retains the descriptive capability of the positive data by enclosing them with a small hypersphere and is able to reject the uncharacteristic patterns. In the meantime, MEMEM exploits the discriminating information provided by the negative samples and therefore is more robust for generalization.

One of the key parameters of MEMEM is $\gamma \in [0, 1]$, which controls the relative weight of the description and discrimination factors considered in the formulation. During the experiments, the best $\gamma$ is chosen by trying different values. For the synthetic data set, the results are similar for a large range of $\gamma$ while for the Biomed data set the choice of $\gamma$ has much more influence on the performance of MEMEM.
CHAPTER 5

RELIABLE ONE-AGAINST-ALL SVM

SVM is originally designed for binary classification. As a margin-based classifier, SVM focuses on the decision boundary instead of directly estimating the conditional probabilities, which, however, makes it difficult to generalize SVM from the binary to multi-class scenarios. This chapter presents a new multi-class learning algorithm named reliable one-against all SVM.

5.1 Introduction

“Single machine” and “error correcting” are two mainstreams for multi-class margin-based classification. As its name suggests, the “single machine” approach attempts to construct a multi-class classifier by solving just a single optimization problem [54, 55, 56, 57, 58]. On the contrary, the “error correcting” approach [59, 60, 61, 62] works with a collection of binary classifiers, for which the primary studies are to determine what binary classifiers should be chosen to train and how to combine their classification results to make the final decision. A good overview of multi-class classification can be found in [63] and [64].

The earliest and one of the most widely used implementations of “error correcting” is the one-against-all method, which constructs $M$ SVM classifiers with the $i$th one
Figure 5.1: A three-class example to illustrate the motivation of considering the classifier’s reliability at the classification stage. (a) Three classes and the true boundaries (solid lines). (b) The linear boundary (the dashed line) that separates class 3 and non-class 3. (c) The linear boundary (the dashed line) that separates class 1 and non-class 1.

separating class $i$ from all the remaining classes. One drawback of this method, however, is that when the $M$ classifiers are combined to make the final decision, the classifier that generates the highest value from its decision function is selected as the winner and the corresponding class label is assigned without considering the competence of the classifiers. In other words, the outputs of the decision function are employed as the only index to indicate how strong a sample belongs to the class. The underlying assumption for doing so is that the classifiers are totally trustable and equally reliable, which does not always hold in multi-class cases.

Fig. 5.1 shows a 3-class example. The solid lines depicted in Fig. 5.1(a) are the true boundaries. Two linear boundaries obtained using the one-against-all approach are shown as the dashed lines in Fig. 5.1(b) and Fig. 5.1(c), respectively. Evidently, the obtained boundary in Fig. 5.1(b) fits exactly the true boundary and therefore the corresponding classifier is more accurate and reliable than that in Fig. 5.1(c). However, they are equally trusted at the classification stage by the one-against-all
method, which may hurt the overall classification accuracy. From this point of view, we speculate that it be advantageous to introduce the discrimination among the SVM classifiers. As will be revealed in Section 5.4, this conjecture finds the theoretical support when we revisit the one-against method in the framework of Dempster-Shafer (D-S) theory [65].

Based on the estimated classification accuracy, we design two measures to quantify the reliability of binary SVM: static reliability measure (SReM) and dynamic reliability measure (DReM). As the name suggests, SReM works in an off-line manner and the result is a constant value regardless of the location of the test sample. DReM, on the other hand, measures the classifier’s reliability in a local region surrounding the test sample. As a result, DReM accounts for the spatial variation of the classifier’s performance but is not as computationally simple as SReM. Based on these two reliability measures, we further propose a new decision strategy for the one-against-all approach to take the classifier’s competence into account. The proposed method has been tested on four UCI data sets and better classification performance has been obtained.

## 5.2 One-Against-All SVM

Consider an $M$-class problem, where we have $N$ training samples: $(x_1, y_1), \ldots, (x_N, y_N)$ with input vector $x_i \in \mathbb{R}^d$ and class label $y_i \in \{1, 2, \ldots, M\}$. The one-against-all approach constructs $M$ binary SVM classifiers, each of which separates one class from all the rest. The $i$th SVM is trained with all the training examples of the $i$th class with positive labels, and all the others with negative labels. Mathematically the $i$th SVM
solves the following problem that yields the ith decision function \( f_i(x) = w_i^T \phi(x) + b_i \):

\[
\begin{align*}
\text{minimize:} & \quad L(w_i, \xi_j^i) = \frac{1}{2} \|w_i\|^2 + C \sum_{j=1}^{N} \xi_j^i \\
\text{subject to:} & \quad \bar{y}_j(w_i^T \phi(x_j) + b_i) \geq 1 - \xi_j^i, \quad \xi_j^i \geq 0,
\end{align*}
\] (5.1)

where \( \bar{y}_j = 1 \) if \( y_j = i \) and \( \bar{y}_j = -1 \) otherwise.

At the classification stage, a sample \( x \) is classified as in class \( i^* \) for which \( f_{i^*} \) produces the largest value

\[
i^* = \arg \max_{i=1,\ldots, M} f_i(x) = \arg \max_{i=1,\ldots, M} (w_i^T \phi(x) + b_i).\] (5.2)

### 5.3 Dempster-Shafer Theory

Dempster-Shafer theory is a theory of evidence that provides a formalism for beliefs (in hypotheses) representation and aggregation. For convenience, this section only recalls the key concepts and leave the details [65] to the interested readers.

Given a set of hypotheses \( \Omega = \{H_1, H_2, \ldots, H_k\} \), let \( 2^\Omega \) be the power set, which consists of the possible \( 2^k \) subsets of \( \Omega \). An element \( \pi \in 2^\Omega \) is a group of hypotheses and is referred as a proposition, whose basic possibility of occurrence is represented by a so-called basic probability assignment (BPA). Formally, a BPA is a function \( m : 2^\Omega \rightarrow [0, 1] \) that satisfies

\[
m(\emptyset) = 0 \quad \text{and} \quad \sum_{\pi \in \Omega} m(\pi) = 1. \] (5.3)

Those \( \pi \)'s that are assigned nonzero probabilities are called focal elements of \( m \).

A belief function, \( \text{Bel}(P) \) of a proposition \( P \) is defined by

\[
\text{Bel}(P) = \sum_{p \in P} m(p). \] (5.4)
Dempster’s rule of combination states that BPAs which correspond to two independent sources of evidence, say, \( m_1 \) and \( m_2 \), can be fused to yield a new BPA \( m \) via

\[
m(P) = m_1 \oplus m_2(P) = K \sum_{P_i \cap P_j = P} m_1(P_i) \cdot m_2(P_j),
\]

(5.5)

where \( K^{-1} = 1 - \sum_{P_i \cap P_j = \emptyset} m_1(P_i)m_2(P_j) \). The operation \( \oplus \) is associative and commutative, and as a result multiple \( m_i \) can be combined sequentially with any arbitrary order.

Lastly, a BPA may be weakened by a positive factor \( a \leq 1 \) to reflect the partial confidence it symbolizes. The scaled BPA is then defined as

\[
m^a(\pi) = \begin{cases} 
a \cdot m(\pi), & \text{if } \pi \neq \Omega; \\
1 - \sum_{\pi \neq \Omega} m^a(\pi), & \text{if } \pi = \Omega.
\end{cases}
\]

(5.6)

### 5.4 Revisiting One-Against-All by Dempster-Shafer Theory

The major difference between the traditional one-against-all method and the proposed one is the way they fuse the outputs from the \( M \) SVM classifiers. Apparently, the fusion rule expressed in Eq. (5.2) is pretty heuristic. However, as we will show in this section, it can be reasoned in the framework of D-S theory. By doing so, the motivation of our new approach can be better understood and deeper insights into its superiority can be obtained.

Again suppose we have a test sample \( x \) and \( M \) one-against-all classifiers \( SVM_i \) with the decision function \( f_i \). In this problem, the set of hypotheses are \( \Omega = \{H_i \} = \{ \text{sample } x \text{ belongs to class } i \} \) with \( i = 1, ..., M \). When each \( SVM_i \) is applied to \( x \), the resulted classification produces a piece of evidence supporting certain hypothesis.

First of all, let us define a BPA function \( m_i \) on \( \Omega \) based on the result of the \( i \)th classification \( SVM_i(x) = \text{sign}(f_i(x)) = y_{fi} \). When \( y_{fi} = 1 \), it is reasonable to increase
our belief in the hypothesis “x belongs to class \(i\)”, and the larger \(f_i(x)\) the stronger the belief. However, this piece of evidence does not provide 100% certainty and therefore only part of it is committed to the hypothesis \(H_i\). The rest of the belief should be given to the proposition \(\Omega\) as a whole because the statement “x does not belong to class \(i\)” does not imply any other hypothesis in particular. The BPA \(m_i\) is therefore defined as

\[
m_i(\pi) = \begin{cases} 
1 - \exp(-|f_i(x)|) = \beta_i, & \text{if } \pi = \{H_i\}; \\
\exp(-|f_i(x)|) = 1 - \beta_i, & \text{if } \pi = \Omega; \\
0, & \text{otherwise.}
\end{cases}
\] (5.7)

Similarly, when \(y_{f_i} = -1\) where the SVM \(i\) classifies \(x\) as not in class \(i\), we define the BPA as

\[
m_i(\pi) = \begin{cases} 
1 - \exp(-|f_i(x)|) = \beta_i, & \text{if } \pi = \{H_i\}; \\
\exp(-|f_i(x)|) = 1 - \beta_i, & \text{if } \pi = \Omega; \\
0, & \text{otherwise,}
\end{cases}
\] (5.8)

where \(\{\bar{H}_i\} = \Omega \setminus \{H_i\} = \{H_1, \ldots, H_{i-1}, H_{i+1}, \ldots, H_M\}\).

After using the Dempster’s rule to obtain the combined BAP \(m = m_1 \oplus m_2 \oplus \cdots \oplus m_M\), the belief function can be computed by Eq. (5.4) and finally the sample \(x\) is classified to the class \(i^*\) with the highest belief. The detailed derivations are given in Appendix A while here we just give the conclusions.

1. If all the SVM \(i\) output generate negative responses,

\[
i^* = \arg \max_{i=1,\ldots,M} \text{Bel}(\{H_i\}) = \arg \min_{i=1,\ldots,M} \beta_i \\
= \arg \min_{i=1,\ldots,M} |f_i(x)| = \arg \max_{i=1,\ldots,M} f_i(x). \tag{5.9}
\]

The last equality holds because all the \(f_i(x)\) are negative.

2. If at least one SVM \(i\) generates positive response,

\[
i^* = \arg \max_{i=1,\ldots,M} \text{Bel}(\{H_i\}) = \arg \max_{f_i(x) \geq 0} \beta_i \\
= \arg \max_{f_i(x) \geq 0} f_i(x) = \arg \max_{i=1,\ldots,M} f_i(x). \tag{5.10}
\]
As one can see, either one of the cases leads to the same decision rule as in Eq. (5.2).

However, as pointed out before, it is not a wise strategy to completely trust the decisions made by the SVM’s. Recall that in D-S theory when the source where the evidences come from cannot be fully trusted, the BPA should be weakened by a belief factor $a$, and this is exactly what the proposed method will do. The question that intermediately follows is what is an appropriate measure to quantify the amount of belief for each SVM. In the next section, we propose the so-called reliability measures and choose them to be the belief factor in Eq. (5.6).

5.5 Reliability Measures For Two-Class SVM

The performance of a classifier is evaluated by the generalization error $R$, which is defined as $R = E[Y = \text{sign}(f(X))]$, where $Y \in \{-1, 1\}$ is the true class label of $X$ and $f$ is the decision function. Obviously, a classifier should be considered more reliable and trustable if it yields smaller $R$ than the other. Unfortunately $R$ is always unknown.

5.5.1 Static Reliability Measure

Using the training error $R_{\text{emp}}$ to estimate $R$ is a straightforward method which has been adopted in many applications. However, as discussed in Chapter 1, when the number of the training samples is relative small with respect to the dimensionality of the feature vector $X$, a small $R_{\text{emp}}$ does not guarantee a small generalization error $R$. An upper bound of $R$ is given in [2, 3, 1], and one advantage of SVM is that minimizing the objective function will also minimize this upper bound [2, 3, 1]. In other words, smaller objective function means smaller generalization error, or a more
reliable classifier. Following this idea, we rewrite the objective function of SVM as

\[
\text{OBJ} = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} (1 - y_i f(x_i))_+ ,
\]

(5.11)

where \((u)_+ = u\) if \(u \geq 0\) and 0 if \(u < 0\), and propose a reliability measure as

\[
\lambda_{\text{SReM}} = \exp \left( -\frac{\text{OBJ}}{\sigma} \right) = \exp \left( -\frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \frac{(1 - y_i f(x_i))_+}{\sigma} \right),
\]

(5.12)

where \(f(x_i) = w^T x_i + b\). The parameter \(\sigma = CN\) is introduced as a normalization factor to offset the effect of the different regularization parameter \(C\) and training size \(N\).

For the linear separable case where \((1 - y_i f(x_i))_+ = 0\) for all the training samples, the measure \(\lambda_{\text{SReM}}\) is reduced to

\[
\lambda_{\text{SReM}} = \exp \left( -\frac{\|w\|^2}{2CN} \right).
\]

(5.13)

Recall that \(\frac{2}{\|w\|^2}\) is the classification margin. The classifier with smaller \(\|w\|\), which corresponds to larger margin, is considered to be more accurate in generalization and therefore its reliability measure \(\lambda_{\text{SReM}}\) is higher.

Note that the test sample \(x\) does not appear in Eq. (5.12) and thus \(\lambda_{\text{SReM}}\) is the same for all the samples. For this reason, it is named static reliability measure (SReM). The computational load of SReM is not high. When the number of support vectors is relatively smaller than the training size \(N\), which happens most of the time, the complexity of SReM is \(O(N)\).

5.5.2 Dynamic Reliability Measure

SReM assumes the classifier to be equally effective throughout the entire feature space. In reality, the classifier’s performance exhibits spatial variation [66] [67], to
accommodate which we extend SReM to DReM, a *dynamic reliability measure*. The basic idea is to estimate the classifier’s reliability in a *local region* of the feature space surrounding the test sample \( x \). Here the local region, denoted as \( N_k(x) \), is defined as the \( k \) nearest neighbors of \( x \) in the training set. Moreover, we are especially interested in the reliability of the classifier with respect to certain output (1 or -1 in this case).

Suppose \( L(x) \in \{1, -1\} \) is the class label assigned to \( x \) by the SVM classifier. Let \( N_k^{L(x)}(x) \) denote the set of the training samples that locate among the \( k \) nearest neighbors of \( x \) and are classified to the same class as \( x \)

\[
N_k^{L(x)}(x) = \{ \hat{x}_j | \hat{x}_j \in N_k(x) \text{ and } L(\hat{x}_j) = L(x) \}.
\]

(5.14)

By rewriting Eq. (5.11) as

\[
OBJ = \sum_{i=1}^{N} \left( \frac{\frac{1}{2}||w||^2}{N} + C(1 - y_i f(x_i))_+ \right) = \sum_{i=1}^{N} OBJ(x_i),
\]

(5.15)

we make the training sample \( x_i \) contributing to the overall OBJ by \( OBJ(x_i) \). In analogy to Eq. (5.11) which takes the summation of \( OBJ(x_i) \) on all samples, we formulate the local version of OBJ as

\[
OBJ_{\text{local}} = \sum_{j=1}^{k_x} OBJ(\hat{x}_j) = \sum_{j=1}^{k_x} \left( \frac{||w||^2}{2N} + C(1 - \hat{y}_j f(\hat{x}_j))_+ \right) = \frac{||w||^2 \cdot k_x}{2N} + C \sum_{i=1}^{k_x} (1 - \hat{y}_j f(\hat{x}_j))_+.
\]

(5.16)

where \( \hat{x}_j \in N_k^{L(x)}(x) \), \( (\hat{x}_j, \hat{y}_j) \) is the training pair, and \( k_x \) is the number of training samples in the set \( N_k^{L(x)}(x) \). Now with \( OBJ_{\text{local}} \) at hand, we can compute the reliability of the decision “\( x \) belongs to \( L(x) \)” by

\[
\lambda_{\text{DReM}}(x) = \exp \left( - \frac{OBJ_{\text{local}}}{C \cdot k_x} \right)
\]

59
\begin{equation}
\exp\left(-\frac{||w||^2}{2CN} + \sum_{i=1}^{k_X}(1 - \hat{y}_j f(\hat{x}_j))_+\right). \tag{5.17}
\end{equation}

From the derivations above we can see that, unlike \( \lambda_{SReM} \), \( \lambda_{DReM}(x) \) is a dynamic function of \( x \), which varies depending on the location of the test sample and the classified label \( L(x) \). For this reason, DReM has to be recomputed as new samples come in and is more expensive than SReM. The extra cost is consumed by finding the \( k \) nearest neighbors, which is \( O(N) \).

### 5.6 A New Decision Rule for One-Against-All

Now with the reliability measures SReM and DReM explained, we are ready for a new decision rule for the one-against-all multi-class classification.

First, we evaluate \( f_i \) at the given sample \( x \) using Eq. (1.22), and generate \( M \) BPA \( m_i \) based on Eq. (5.7) and (5.8). Considered as an independent source of information, the evidence produced by the classifier SVM\(_i\) obviously should be granted more belief if SVM\(_i\) is more reliable. Here we employ the reliability measures directly as the belief factor in Eq. (5.6) to reduce the original BPA \( m_i \). All of the derivations in Section III preserves except for the substitution of \( \beta_i \) with \( \lambda_{SReM}\beta_i \) or \( \lambda_{DReM}\beta_i \) depending on which measure is adopted, and then we reach the following decision rules:

1. All the SVM\(_i\) generate output negative responses.

\begin{equation}
\hat{i}^* = \arg \min_{i=1,...,M} \lambda_i \beta_i = \arg \min_{i=1,...,M} \lambda_i (1 - \exp^{-|f_i(x)|}). \tag{5.18}
\end{equation}

2. At least one SVM\(_i\) generates positive response.

\begin{equation}
\hat{i}^* = \arg \max_{f_i(x) \geq 0} \lambda_i \beta_i = \arg \max_{f_i(x) \geq 0} \lambda_i (1 - \exp^{-|f_i(x)|}). \tag{5.19}
\end{equation}
Here $\lambda_i$ denotes the SReM or DReM of SVM$_i$. As a matter of fact, the two rules listed above can be unified into one as

$$i^* = \arg \max_{i=1,...,M} \lambda_i \cdot \{ \text{sign}(f_i(x))(1 - \exp^{-|f_i(x)|}) \}.$$

Eq. (5.20) is an very interesting result. The quantity $y_{f_i}$, which varies between -1 and 1, can be thought as a soft decision that carries two kinds of information: the sign part encodes the hard decision on “$x$ belongs to class $i$ or not” and its absolute value represents how strong the decision is. When $\lambda_i$ are equal (including $\lambda_i = 1$), which implies the classifiers are equally competent, Eq. (5.20) will reduce to Eq. (5.2), the conventional version of the one-against-all classification.

### 5.7 Experimental Result

The proposed approach has been applied to the multi-class data sets obtained from the UCI repository of machine learning [68], and demonstrates its advantage over the conventional counterpart. This section reports the experimental results on four data sets that exhibit certain varieties, i.e., the number of classes to be differentiated and the kernel function to be used. For the training of each SVM, the generalization accuracy is estimated by testing different values of $C \in [2^{12}, 2^{11}, ..., 2^{-2}]$, and the best one is chosen for the performance comparison.

The first data set is the image segmentation data. Each sample has 19 continuous attributes, which are collected from a 3 x 3 region of an outdoor image. There are seven classes to classify: brickface, sky, foliage, cement, window, path, and grass. The training set consists of 210 samples with 30 per class while the size of the test set is 2100 with 300 samples per class. The polynomial kernel with $d = 1$, which has
been reported as a good choice for the image segmentation data [69], is adopted as the kernel function. Fig. 5.2(a) shows the classification errors yielded by using the decision functions \( f(x) \) (the conventional one-against-all method), SReM and DReM, respectively. The errors are plotted by including the classes one by one (in alphabet order). As one can see, SReM and DReM always lower the error percentage and DReM performs the best.

The second experiment is conducted on wine recognition data set. The data contains the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. It provides 178 samples in total, which is distributed into 3 classes. Each sample has 13 continuous attributes. Linear SVM is trained using 128 samples that are randomly chosen from the set and its performance is tested on the rest 50 instances. The process is repeated for 100 times, and the averaged errors is plotted in Fig. 5.2(b).

The iris plant data set is a small set yet one of the best-known data sets to be found in the pattern recognition literature. It contains 3 classes of 50 instances each, where each class refers to a type of iris plant. The class *iris setosa* is linearly separable from *iris Versicolour* and *iris Virginica* while the latter two are linearly nonseparable. Similar to the previous data set, linear SVM is adopted. The classification errors, as shown in Fig. 5.2(c), is obtained by leave-one-out cross validation.

The last data set is the letter recognition data. This data set contains 20000 samples, each of which corresponds to one of the 26 capital letters in the English alphabet. 16 integer-valued features such as statistical moments and edge counts are provided to represent each letters. Typically the first 16000 samples are used as the training data and the remaining 4000 as the test data. After experimenting with
Table 5.1: Comparison of classification errors (Boldface indicates the best performance).

<table>
<thead>
<tr>
<th></th>
<th># of classes</th>
<th># of attributes</th>
<th>classification errors (%)</th>
<th>error reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>f(x)</td>
<td>SReM</td>
<td>DReM</td>
<td>SReM</td>
</tr>
<tr>
<td>Image Segmentation</td>
<td>7</td>
<td>19</td>
<td>9.24</td>
<td>8.57</td>
</tr>
<tr>
<td>Wine Recognition</td>
<td>3</td>
<td>13</td>
<td>3.79</td>
<td>3.63</td>
</tr>
<tr>
<td>Iris Plant</td>
<td>3</td>
<td>4</td>
<td>4.0</td>
<td>2.67</td>
</tr>
<tr>
<td>Letter Recognition</td>
<td>26</td>
<td>16</td>
<td>3.98</td>
<td>3.97</td>
</tr>
</tbody>
</table>

different kernel functions, the RBF is found to be the best choice for this 26-class problem. Yielding a total of 3.98% misclassifications, all the 26 classifiers perform well, and therefore SReM and DReM are only able to reduce the errors to 3.97% and 3.77% respectively as shown in Fig. 5.2(d) and Table 5.1.

### 5.8 Summary

One-against-all is a conventional yet widely-used method to extend SVM from the binary to multi-class classification. The decision strategy of one-against-all, which treats all the constructed SVMs equally, is under study in this chapter. We argue intuitively as well as theoretically that it be advantageous to introduce discrimination to the stage when classification results of multiple SVMs are combined. To address this problem, two quantities, SReM and DReM, are proposed to measure the reliability of SVM based on the estimated generalization accuracy. Utilizing the reliability measures, we further suggest a new decision strategy for the one-against-all method. The experimental results have demonstrated that the proposed approach is able to improve the classification accuracy without imposing high computational cost.
Figure 5.2: Comparison of the performance by including the classes one by one. (a) Image segmentation data set using 1-degree polynomial as the kernel function. (b) Wine recognition data set using the linear kernel function. (c) Iris plant data set using the linear kernel function. (d) Letter recognition data set using RBF as the kernel function
CHAPTER 6

SVM AND FEATURE SELECTION

In many pattern recognition applications, high-dimensional feature vectors impose a high computational cost as well as the risk of “overfitting”. Feature Selection addresses the dimensionality reduction problem by determining a subset of available features which is most essential for classification. This chapter presents a novel feature selection method named Filtered and Supported Sequential Forward Search (FS_SFS) in the context of SVM.

6.1 Introduction

Reduction of feature dimensionality is of considerable importance in machine learning. The reason for being so is twofold: to reduce the computational complexity and to improve the classifier’s generalization ability. The first motivation is quite evident since fewer features require less run time to train and to apply the classifier. The second motivation is low dimensional representation reducing the risk of “overfitting”. As a rule of thumb, a minimum of $10 \cdot d \cdot M$ training samples is required for a $d$-dimensional classification problem of $M$ classes [70]. When it is impractical
and even impossible to obtain the required number of training samples, the reduc-
tion of feature dimensionality helps decrease the size of the training samples and consequently improves the generalization performance of the classification algorithm.

Feature extraction and feature selection are two different approaches for the reduc-
tion of dimensionality. Feature extraction involves linear or nonlinear transformation from the original feature space to a new one of lower dimensionality. Although it does reduce the dimensionality of the vectors fed to the classifier, the number of features that must be measured remains the same. Feature selection, on the other hand, directly reduces the number of original features by selecting a subset of them that still retains sufficient information for classification. Feature selection techniques have been applied successfully in many applications, such as automated text categoriza-
tion [71] and data visualization [72]. In general, feature selection approaches can be grouped into two categories: filter methods and wrapper methods [73]. Acquiring no feedback from classifiers, the filter methods estimate the classification performance by some indirect assessments such as distance measures which reflect how well the classes separate from each other. The wrapper methods, on the contrary, are classifier-dependernt. Based on the classification accuracy, the methods evaluate the “goodness” of the selected feature subset directly, which should intuitively yield better performance. As a matter of fact, many experimental results reported so far are in favor of the wrapper methods [73, 74, 75].

In spite of the good performance, the wrapper methods have limited applications due to the high computational complexity involved. This is true especially when the wrapper methods are applied to SVM. Given the fact that training SVM even only once needs a great deal of computation when the number of training samples is large,
the integration of SVM and wrapper methods, which requires multiple times of SVM training, will be computationally infeasible. Even when some well-known suboptimal search strategies such as the sequential forward search (SFS) are used, the selection process is still quite costly. That calls for feature selection methods designed especially for SVM. Unfortunately there are just a few algorithms in the literature that have been proposed for feature selection in the context of SVM [76, 77, 78, 79, 80]. One possibility is to embed feature selection into the optimization process [76][77]. For example, [76] adds an extra term that penalizes the size of the selected feature subset to the standard cost function of SVM, and optimizes the new objective function to achieve feature selection. A similar idea is also employed in [77]. The major difference is that [77] introduces a binary vector whose elements indicate the presence and absence of the corresponding feature component, and then approximates the binary vector with a nonnegative real-valued vector $\sigma$ so that optimization can be performed efficiently via gradient descent. Then the features corresponding to the $m$ largest valued elements of $\sigma$ are selected. The benefit is that optimization has to be done only once. Unfortunately the two methods evaluate the features on a individual basis and the correlation between them is ignored [79].

In the meantime, many researchers suggest to evaluate the importance of features by measuring the change of the cost function when a feature is removed, or equivalently when its weight is set to zero. In the case of SVM which has a quadratic cost function, the magnitude of weights $w_i$ is a reasonable feature ranking criterion, and based on this criterion a SVM recursive feature elimination (SVM RFE) method is proposed in [78]. However, this approach is limited to linear kernels. Some researchers propose to utilize the change of the discriminant function rather than the
cost function itself for feature ranking [79][80]. Since for most kernels the discriminant function of SVM is differentiable with respect to individual features, the algorithms become applicable to nonlinear kernels.

In this chapter we present a more efficient version of the wrapper/SFS method for SVM which is named Filtered and Supported Sequential Forward Search (FS_SFS). FS_SFS is designed especially for SVM and has the following properties to improve its efficiency over the conventional wrapper/SFS method:

1. **FS_SFS combines the advantages of the filter and the wrapper methods.** By introducing a filtering process for each SFS iteration, FS_SFS reduces the number of features that has to be tested through the training of SVM. Then the pre-selected features are considered “informative” and are evaluated by the accuracy of classification as in the conventional wrapper method. In this way, we are able to reduce the unnecessary computation time spent on the testing of the “noninformative” features while maintaining the good performance delivered by the wrapper method.

2. **FS_SFS introduces a new criterion that assesses features in a collectively manner.** An effective filtering criterion is needed in FS_SFS since it is undesirable to discard many informative features through the filtering process. To address this problem, we develop a new criterion, which is computationally efficient and considers the discriminant ability of individual features as well as the correlation between them.

3. **FS_SFS is specially designed for SVM classifier to improve the efficiency of the feature selection process.** During the feature search process,
FS_SFS dynamically maintains an active set for training, which is a subset of the original training samples, as the candidates of the support vectors. Whenever the training of SVM is needed, only the samples in the subset are utilized. In this way the efficiency for training a single SVM classifier is improved.

6.2 FS_SFS: Filtered and Supported Sequential Forward Search

6.2.1 Problem Statement of Feature Selection

Consider the binary classification scenario which has input vectors denoted as \( X \in \mathbb{R}^d \) and their corresponding class labels denoted as \( Y \in \{1, -1\} \). Let

\[
F = \{f_1, f_2, \ldots, f_d\}
\]

(6.1)

be the set of all features under examination, and let

\[
Z = \{(x_l, y_l) \mid l = 1, 2, \ldots, N\}
\]

(6.2)

\[
= \{[x_l^1 \, x_l^2 \, \ldots \, x_l^d]^T, y_l) \mid l = 1, \ldots, N\}
\]

denote the training set containing \( N \) training pairs, where \( x_l^i \) is the numerical value of feature \( f_i \) for the \( l \)th training sample.

The goal of feature selection is to find a minimal set of features \( F_s = \{f_{s1}, f_{s2}, \ldots, f_{sk}\} \) to represent the input vector \( X \) in a lower-dimensional feature space as:

\[
X_s = [x^{s1} \, x^{s2} \, \ldots \, x^{sk}],
\]

(6.3)

where \( k < d \), while the classifier obtained in the low-dimensional representation still yields acceptable classification accuracy.
Figure 6.1: Comparison between (a) filter methods, (b) wrapper methods, (c) and the proposed methods for feature selection.
6.2.2 Review of FS_SFS

As mentioned earlier, feature selection approaches can be categorized into two classes: the filter and the wrapper methods [73], whose pipelines are shown in Fig. 6.1(a) and 6.1(b), respectively. Guided by the feedback from the classifier, the wrapper method selects features in a more consistent way than the filter methods. The better performance of the former is achieved, however, at the cost of much more computation. Fig. 6.1(c) gives the outline of the proposed method which combines the advantage of both the filter and wrapper methods. The filtering part, acting in the generic way similar to a filter method, ranks features without involving the classifier. The features with relatively high ranks are considered as “informative” feature candidates and then are re-studied by the wrapper part that further investigates their contributions to a specific classifier. This combinational framework delivers as good a performance as the conventional wrapper method but is computationally simpler.

With the framework determined, feature selection is reduced to a problem of searching for the optimal subset [81]. Many search strategies have been proposed [82, 83, 84], from which we adopt a suboptimal search method called sequential forward search (SFS) [84] for its simplicity and effectiveness. Starting with an empty set, SFS iteratively selects one feature at a time and add it to the current feature set. The feature added is the one which gives the smallest value according to a certain criterion comparing to adding the other remaining features. Different approaches have different criteria, such as class separability [85], classification errors [86], change of the boundary [87] and etc. The criterion employed in our algorithm is the objective function of SVM formulated in Eq. (2.7).
The FS_SFS method will be presented in detail in the following three subsections. The isolated filter and wrapper parts, which are named Filtered_SFS (F_SFS) and Supported_SFS (S_SFS) respectively, are presented in Subsection 6.2.3 and 6.2.4. Then they are then integrated as FS_SFS in Subsection 6.2.5.

6.2.3 F_SFS: Filtered_SFS Using a New Criterion

Recall that the goal of the filter part is to discard some “noninformative” features to reduce the computational burden of the wrapper part. To serve this purpose, the filter part needs to meet two major requirements:

- the criterion for filtering must be simple so as not to introduce much computation;

- the process of filtering should lose as few informative features as possible.

Class separability is a classical criterion of filtering available in the literature. It involves calculating the normalized distance between classes and then eliminating the features that yield low separability values. The criterion is computationally simple and thus satisfies the first requirement. However, it has a major drawback, i.e., the criterion implicitly assumes the features to be orthogonal and overlooks the correlation between them. Consequently those correlated features that individually separate the classes well but collectively provide redundant information might be retained, which violates the second requirement.

Here we propose a new criterion which is able to yield a more compact feature subset. In this new criterion, the correlation between features as well as the class separability of individual features are taken into consideration. Also it retains the advantage of simple computation.
Suppose we have a feature combination \( F_s = \{ f_{n_1}, f_{n_2}, \ldots, f_{n_k} \} \), and we can calculate a score for each individual feature \( f_i \). This score, which is denoted as \( R_{i,F_s} \), is the measure of the importance of that particular feature \( f_i \) such that the higher the score the more important the feature is. The evaluation of the score for a given feature subset \( F_s \) takes the following steps.

1. **determining the discriminant ability of the feature**

   The discriminant ability of feature \( f_i \) is described by the class separability as
   \[
   D_i = \frac{|m_i^1 - m_i^2|}{std_i^1 + std_i^2},
   \]
   (6.4)
   where \( m_i^1 \) and \( std_i^1 \) (\( m_i^2 \) and \( std_i^2 \)) are the mean and standard deviation of the samples belonging to class 1 (-1) when only feature \( f_i \) is considered. It can be seen from Eq. (6.4) that the further the two classes are separated from each other using \( f_i \), the larger \( D_i \) would be, and therefore the better discriminant ability feature \( f_i \) has.

2. **determining the correlation between \( f_i \) and \( F_s \)**

   First we define the correlation coefficient \( \rho_{i,j} \) between two features \( f_i \) and \( f_j \) as:
   \[
   \rho_{i,j} = \prod_{c=1}^{2} \rho_{i,j}^{(c)}
   = \prod_{c=1}^{2} \frac{\text{cov}(S_c(f_i), S_c(f_j))}{\sqrt{\text{var}(S_c(f_i)) \cdot \text{var}(S_c(f_i))}},
   \]
   (6.5)
   where \( S_c(f_i) = \{ x_i^c | y_i = c \} \) is the training vectors that are represented by feature \( f_i \) and labeled as class \( c \).
Based on $\rho_{i,j}$, we further define the correlation coefficient between a single feature $f_i$ and a feature set $F_s$ as

$$\rho_{i,F_s} = \max_{f_j \in F_s} |\rho_{i,j}|.$$  \hspace{1cm} (6.6)

A high value of $\rho_{i,F_s}$ indicates that $f_i$ is highly correlated with certain feature $f_j \in F_s$ and therefore it carries redundant information.

3. **calculating the score of the feature**

It is desirable to select the features that can individually separate the classes well and has small correlation with the features in the subset which has been obtained so far. Thus the final score assigned to $f_i$ is defined as:

$$R_{i,F_s} = \frac{D_i}{\max\{D_i\}} - |\rho_{i,F_s}|,$$  \hspace{1cm} (6.7)

where $D_i$ is normalized such that it is in the same range as $|\rho_{i,F_s}|$.

It is worth noting that $R_{i,F_s}$ is dependent not only on $f_i$ but also on $F_s$. As a result, the score of $f_i$ usually changes during the SFS process.

### 6.2.4 S_SFS: Supported_SFS in the Context of SVM

Supported SFS is basically a variation of the SFS algorithm that is specially tailored to SVM to expedite the feature searching process. Recall that in SVM there is a special group of training samples named “support vectors”, whose corresponding coefficients $\alpha_i$ in Eq. (1.19) are non-zeros. In other words, training samples other than support vectors have no contribution to determining the decision boundary. Since the number of support vectors is relatively small, we could train SVM just by using the support vectors. Following this idea, we propose the supported SFS algorithm, which
dynamically maintains an *active subset* as the candidates of the support vectors, and trains SVM using this reduced subset rather than the entire training set. In this way, we are able to find the boundary with less computational cost.

The procedure of S_SFS is described as follows. The first step is to choose the best single features among the $d$ possible choices. To do so, we train SVM $d$ times, each of which uses all the training samples available but with only one feature $f_i$. Mathematically the initial feature combination set is

$$F^i_1 = f_i, f_i \in F, \quad (6.8)$$

and the active training set $V^i_1$, which is the entire training set, is

$$V^i_1 = \{1, 2, \ldots, N\}. \quad (6.9)$$

Although every training sample in $S$ is involved in this initial training task, the computational complexity is not high because the input vector is just one-dimensional. After the training, each single-feature combination $F^i_1$ is associated with a value $M^i_1$, which is the minimum of the objective function, and a group of support vectors $v_i$. The feature that yields the smallest $M^i_1$

$$j = \arg \min_{i \in \{1, 2, \ldots, N\}} M^i_1 \quad (6.10)$$

is chosen as the best one. Thus we obtain the initial feature combination $F_1 = \{f_j\}$ and its active training set $V_1 = \{v_j\}$.

At step $n$, we have already obtained the feature combination $F_n$ that contains $n$ features, and the active training set $V_n$. To add one more feature into the feature combination set, we test each remaining feature $f_i$ one by one and construct the
corresponding active training set for every new feature combination as follows:

\[
\begin{align*}
F^i_{n+1} &= F_n \cup \{f_i\}, \text{ for } f_i \in F^\text{av}_n, \\
V^i_{n+1} &= V_n \cup \{v_i\},
\end{align*}
\]

(6.11)

where \(F^\text{av}_n = \{f_r \mid f_r \in F \text{ and } f_r \not\in F_n\}\) is the collection of the available features to be selected from.

For each \(F^i_{n+1}\) we train SVM just by using the samples in \(V^i_{n+1}\). The resulting minimum of the objective functions and the collection of the support vectors are denoted as \(M^i_{n+1}\) and \(SV^i_{n+1}\), respectively. Then the feature that yields the combination with the least \(M^i_{n+1}\)

\[
j = \arg \min_{f_i \in F^\text{av}_n} M^i_{n+1}
\]

(6.12)
is selected, and accordingly the new feature combination \(F_{n+1}\) and new active training set \(V_{n+1}\) are obtained as follows:

\[
\begin{align*}
F_{n+1} &= F^j_n, \\
V_{n+1} &= SV^j_{n+1}.
\end{align*}
\]

(6.13)

The SFS process continues until no significant reduction of \(M^j_n\) is found or the desired number of features has been obtained.

6.2.5 FS_SFS: the Integration of F_SFS and S_SFS

The integration of F_SFS and S_SFS is quite straightforward for which the basic idea is to discard the features with low scores according to the criterion discussed in Subsection 6.2.3 so as to reduce the number of features which S_SFS has to evaluate.

Assuming we are at step \(n\) of SFS with \(F_n\) and \(V_n\) available, FS_SFS works as follows.

1. calculate the score \(R_{i,F_n}\) for each remaining feature \(f_i\);
2. select $K_n$ highest scored features to construct $F_n^{\text{av}}$;

3. determine the next feature to be added using Eq. (6.11) and Eq. (6.12);

4. update the active training set using Eq. (6.13).

$K_n$ determines how many features we want to keep after the filtering at the step $n$. One extreme case is that $K_n$ is equal to the number of all remaining features. In this scenario the filter part does not contribute at all and evidently FS_SFS is reduced to S_SFS. Similarly, if $K_n$ is equal to 1, the wrapper method is unnecessary and FS_SFS becomes F_SFS. $K_n$ is usually chosen between the two extremes and thus works as a tuning parameter to balance between the performance and the complexity of the algorithm.

### 6.3 Experimental Results

In the experiments, we apply the proposed feature selection method to both synthetic and real-world data. First, we design a synthetic data set to test whether the support vectors are effectively chosen by the active training set. Then we adopt the data set in [77] to compare the performance of FS_SFS and other three algorithms, which demonstrates that FS_SFS is more capable of selecting a small number of features when most of the available features are irrelevant. Finally, we employ 10 data sets which are from the widely-used UCI repository of machine learning [68] to test the capability of FS_SFS on real-world problems. For all the experiments, the optimization of SVM is achieved by SVM Torch [88].
6.3.1 Results on Synthetic Data

Three experiments are carried out on a synthetic data set. For each experiment we use \( N \) vectors \( X = (x^1, x^2, \ldots, x^d) \) from two classes (class 1 or class -1) in a \( d \)-dimensional data space. The components \( x_i \) are independent Gaussian variables whose distributions are designed as follows:

\[
p(x^i) = \begin{cases} 
\frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(\frac{x_i - 1}{2\sigma_i^2}\right), & \text{if } X \text{ belongs to class 1;} \\
\frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(\frac{x_i + 1}{2\sigma_i^2}\right), & \text{if } X \text{ belongs to class -1,}
\end{cases}
\]  

(6.14)

where \( \sigma_i = 0.5 \cdot 2^{(i-1)} \) and \( i = 1, 2, \ldots, d \).

The three experiments deal with 2-dimensional, 3-dimensional and 10-dimensional data, respectively. The values of \( N \) and \( d \) in each experiment are:

1. 2-dimensional case: \( N = 100 \) and \( d = 2 \);
2. 3-dimensional case: \( N = 100 \) and \( d = 3 \);
3. 10-dimensional case: \( N = 250 \) and \( d = 10 \).

Fig. 6.2 shows the effectiveness of the FS_SFS algorithm in estimating the support vectors in the 2-D scenario. By training SVM using only \( x^1 \), we obtain the support vectors associated with \( x^1 \), which are denoted as \( v_1 \) and circled in Fig. 6.2(a). Similarly, we obtain \( v_2 \) (Fig. 6.2(b)). Then as discussed in Section III, FS_SFS trains SVM by using only samples \( V = v_1 \cup v_2 \). As one can see from Fig. 6.2(c) and Fig. 6.2(d), FS_SFS yields exactly the same support vectors as the standard SVM training method which involves all the original training samples.

We also test FS_SFS for the 3-D case, and Fig. 6.3 shows how the active training set changes when more and more features are added to the candidate feature set \( F \). Again, FS_SFS and the standard SVM methods generate the same support vectors.
Figure 6.2: The active training set of the 2-D case, which are circled, maintained by S_SFS. (a) $v_1$, which is the support vectors obtained by considering only feature $x^1$. (b) $v_2$, which is the support vectors obtained by considering only feature $x^2$. (c) The support vectors obtained by training SVM on $V = v_1 \cup v_2$. (d) The support vectors obtained by using all the training samples.
Figure 6.3: The active training set of the 3-D case, which are circled, maintained by $S_{SSFS}$. (a) The supported vectors obtained when $F = \{x^1\}$. (b) The supported vectors obtained when $F = \{x^1, x^2\}$. (c) The supported vectors obtained when $F = \{x^1, x^2, x^3\}$. (d) The support vectors obtained by using all the training samples.
The third experiment is carried out on the 10-D feature space, and $K_n$ is set to

$$K_n = \left\lfloor \frac{|F_n|}{2} \right\rfloor,$$

where $|F_n|$ denotes the number of features in the feature set $F_n$. In other words, half of the available features are discarded at every SFS iteration step. According to Eq. (6.14), the samples are generated in such a way that if $i < j$ the variance of feature $x^i$ is larger than that of $x^j$, and therefore $x^i$ has more discriminant ability than $x^j$. For that reason, we expect $x^i$ to be selected before $x^j$. For the convenience of display, we assign a feature $x^i$ a point as the following:

$$\text{Point}(x^i) = 11 - \text{pos}(x^i),$$

where $\text{pos}(x^i)$ is the order of $x^i$ selected. For example, if $x^i$ is the number one feature selected, its point would be 10. Fig. 6.4(a) gives the ideal point of $x^i$. Fig. 6.4(b) and Fig. 6.4(c) show the actual points of the features, which are averaged over 100 trials, when SFS and FS_SFS are applied, respectively. Here by notation SFS, we mean the
<table>
<thead>
<tr>
<th>$K_n$</th>
<th>classification accuracy</th>
<th>run time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>training</td>
<td>testing</td>
</tr>
<tr>
<td>$\lfloor \frac{</td>
<td>F_n</td>
<td>}{4} \rfloor$</td>
</tr>
<tr>
<td>$\lfloor \frac{</td>
<td>F_n</td>
<td>}{2} \rfloor$</td>
</tr>
<tr>
<td>$\lfloor \frac{3}{4} \frac{</td>
<td>F_n</td>
<td>}{4} \rfloor$</td>
</tr>
<tr>
<td>$</td>
<td>F_n</td>
<td>$</td>
</tr>
</tbody>
</table>

Table 6.1: Comparison of classification accuracy and run time with different values of $K_n$. 

wrapper methods using the SFS strategy. As one can see, FS_SFS is able to achieve similar results as SFS but with a lower computational cost.

Different values of the parameter $K_n$ are tried for the 10-D case, and the classification accuracy and run time obtained for the case are listed in Table 6.1. Not to our surprise, with the increasing of $K_n$ the accuracy of the classification increases but the selection process takes longer time, which confirms that the performance and complexity of the algorithm can be balanced by tuning $K_n$.

### 6.3.2 Results on Synthetic Data 2

In the experiment described above, all the features to be selected from are more or less relevant. In order to test the performance in the presence of a large number of irrelevant features, we adopt the artificial data designed in [77] in which only 6 out of total 202 available features are useful for the classification. The data set is constructed such that the class labels are evenly distributed as $P\{y = 1\} = P\{y = -1\} = 0.5$. The feature vectors $X$ are 202-dimensional which are sampled according to the probability distribution function (pdf) shown in the following equation such
that only the first 6 are relevant while the rest of them is just noise.

\[
P\{x^i|y\} = \begin{cases} 
0.7 \cdot y \cdot N(i, 1) + 0.3 \cdot N(0, 1), & \text{if } 1 \leq i \leq 3; \\
0.3 \cdot y \cdot N(i - 3, 1) + 0.7 \cdot N(0, 1), & \text{if } 4 \leq i \leq 6; \\
N(0, 1), & \text{if } 7 \leq i \leq 202. 
\end{cases} 
\]  

(6.17)

FS_SFS is applied to this data set to select the best 2 features, and Fig. 6.5 gives the average classification errors on 500 testing samples versus various training set sizes. The performance of the standard SVM as well as three other feature selection algorithms, which are DFPA [79], SVM BFE [78], and the classical filter method using class separability as the filtering criterion, respectively, are also presented for the purpose of comparison.

As one can see, the presence of a large number of irrelevant features does hurt the performance, which again demonstrates the importance of feature selection. FS_SFS outperforms the other algorithms, especially when the training size is small. As more
<table>
<thead>
<tr>
<th>name of data sets</th>
<th>abbreviation</th>
<th># of features</th>
<th># of instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUPA Liver Disorders</td>
<td>BUPA Liver</td>
<td>6</td>
<td>354</td>
</tr>
<tr>
<td>Wisconsin Breast Cancer</td>
<td>BCW</td>
<td>9</td>
<td>683</td>
</tr>
<tr>
<td>‘A’ and ‘B’ of Letter Image Recognition</td>
<td>A-B Letter</td>
<td>16</td>
<td>1555</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>Ionosphere</td>
<td>34</td>
<td>351</td>
</tr>
<tr>
<td>Glass Identification</td>
<td>Glass</td>
<td>9</td>
<td>214</td>
</tr>
<tr>
<td>Heart Disease</td>
<td>Heart Disease</td>
<td>13</td>
<td>303</td>
</tr>
<tr>
<td>Pima Indians Diabetes</td>
<td>PI Diabetes</td>
<td>8</td>
<td>768</td>
</tr>
<tr>
<td>Japanese Credit Screening</td>
<td>Credit Screening</td>
<td>15</td>
<td>690</td>
</tr>
<tr>
<td>Postoperative Patients</td>
<td>PO Patients</td>
<td>8</td>
<td>90</td>
</tr>
<tr>
<td>Wisconsin Diagnostic Breast Cancer</td>
<td>WDBC</td>
<td>30</td>
<td>569</td>
</tr>
</tbody>
</table>

Table 6.2: Ten data sets from UCI

and more training samples are used, SFS_SFS performs marginally better than the method of DFPA [79].

6.3.3 Results on Real-World Data

The proposed algorithm is applied to ten real-world data sets [68], the detailed information of which is listed in Table 6.2. For each data set we randomly set aside 20% instances as the testing samples, and the rest as the training samples. Again we let $K_n = \left\lfloor \frac{|F_n|}{2} \right\rfloor$. Fig. 6.6 shows the performance of FS_SFS on the A-B-letter data set. The solid line in Fig. 6.6(a) represents the value of the objective function obtained by FS_SFS. As one can see, it monotonically decreases with more features added and gradually converges to the dash-dot line, which depicts the value of the objective function when SVM is trained utilizing all the training samples and all the available features. Fig. 6.6(b) plots the classification errors yielded by FS_SFS and SFS, respectively, where they yield comparable errors over both the training and the testing sets. The similar results are observed for the other nine data sets, and Fig. 6.7 shows the performance comparison on three data sets.
Figure 6.6: The results of letter recognition (A and B). (a) The change of the value of the objective function vs. the number of retained features. (b) The comparison of classification errors between FS_SFS and SFS.

Figure 6.7: Comparison of classification errors between FS_SFS and SFS. (a) The Wisconsin Breast Cancer data set. (b) The Johns Hopkins University Ionosphere data set. (c) The Glass Identification data set.
Table 6.3: Comparison of the average classification accuracy and run time between FS,SFS and SFS over 20 trials. The range of the number of features selected are given in the parentheses as (min, max). For BCW, Glass and Post-operative Patients data set, the linear SVM is employed. For the rest, nonlinear SVM is utilized and the radial basis function is adopted as the kernel function.

When the value of the objective function does not decrease significantly, the feature selection process stops and the features that have not been selected at that point are considered irrelevant to the classification problem. Throughout the experiments, FS,SFS and SFS always select the same number of features when the stop condition is satisfied. More detailed results are listed in Table 6.3, which evidently shows that FS,SFS improves the efficiency of SFS without sacrificing the accuracy of either the selection or the classification.

FS,SFS also shows stability in selection of features. An example is shown in Fig. 6.8(a), in which X and Y axes are the indices of features and trials respectively while the selected features are highlighted by squares. For example, in the first trial 5 features (feature #1, #4, #6, #7 and #8) are selected, and in the fifth trial 6 features (feature #1, #3, #4, #6, #7 and #8) are selected. Over the 20 trails conducted, the results are stable: feature #1, #4, #6, #7 and #8 are always selected while feature...
#5 and #9 are not. The stability can be seen more clearly in Fig. 6.8(b) which displays the total times each feature has been selected over the 20 trials.

As discussed before, one novelty of FS_SFS is that the repeated training of SVM is conducted on the subsets of the original training sets. In order to concretely show how this strategy helps reduce the computational cost, in Fig. 6.9 against the number of the selected features we plot the change of \( R_a \), which is defined as

\[
R_a = \frac{S_a}{S_i} = \frac{\# \text{ of the samples in the subsets}}{\# \text{ of the samples in the original training sets}}.
\] (6.18)

As one can see, although the value of \( R_a \) varies from data set to data set, it does not grow quickly during the selection process. Actually, it stays much less than 1 except for the first iteration when the active training set \( V^i_1 \) is the entire set (Eq. (6.9)) and therefore \( R_a = 1 \). It is also observed that \( R_a \) hardly decreases significantly as the iteration process continues and here we offer an explanation. According to Eq. (6.11), we know that

\[
|V^i_{n+1}| = |V_n| + |v_i| - |V_n \cap v_i| \geq |v_i|,
\] (6.19)

where \(| \cdot |\) denote the number of elements in the corresponding set. More specifically, the average size of the active training set is lower bounded by

\[
\frac{\sum_{i \in F^a_n} |v_i|}{|F^a_n|},
\] (6.20)

which prevents the decrease of \( R_a \).

Table 6.4 gives the performance comparison of FS_SFS, DFPA [79], SVM BFE ([78] for linear and [80] for nonlinear problems) and the filter method using class separability as the filtering criterion, respectively. Similar to what we have observed on the synthetic data 2, when the number of training samples is insufficient with
Figure 6.8: The stability of FS_SFS on the BCW data set. (a) The selected features (squared) during trials. (b) Each bar shows how many times the corresponding feature are selected over 20 trials.

![Graphs showing feature selection stability on BCW data set.](image)

Figure 6.9: The change of $R_a$, which is defined as the ratio of the size of the subset (used for training) to the size of the total training set, during the feature selection process for six data sets.

![Graphs showing the change of $R_a$ for different data sets.](image)
Table 6.4: Comparison of the classification accuracy between FS_SFS, DFPA [79], SVM BFE ([78] for linear and [80] for nonlinear problems) and filter method using class separability as the filtering criterion (Boldface indicates the best performance).

respect to the number of features, which is the case for the Post-Operative Patient data set (90 instances with 8 features), FS_SFS achieves significantly higher classification accuracy than the other three approaches. Another special data set is the Credit Screening data set. The instances in this set exhibit a good mix of attributes – continuous, nominal with small numbers of values, and nominal with larger numbers of values, and in this case FS_SFS again shows major advantage. For the rest of the adopted data set where most of the features are relevant and the training samples are relatively ample, the performances of FS_SFS, DFPA and SVM BFE are close. Nevertheless, FS_FSF still yields the best results most of the time.

6.4 Summary

In this chapter, we present a novel feature selection method in the context of SVM, which is fundamentally a more efficient version of the wrapper/SFS approach. As SVM is becoming a popular classifier and the inefficiency of slow training has been a bottleneck of many applications using SVM, the contribution of this work is
important for significantly reducing the time of training. The experimental results show that the proposed method delivers as good a performance when the features are more or less relevant and produces a more effective classifier when many irrelevant features are present.
CHAPTER 7

CLASSIFICATION-BASED VIDEO OBJECT EXTRACTION USING SOFT SVM AND $\psi$-LEARNING

VO extraction refers to the process that locates VOs, which may be arbitrary collections of images regions with non-rigid motion, with pixel-wise accuracy. We consider VO extraction as a classification problem. Single VO extraction, for example, requires identifying each pixel as either foreground or background. Similarly, multiple VO extraction can be formulated as a multi-class classification problem. From this perspective, we propose a novel semiautomatic approach for VO extraction, which is significantly different from conventional approaches yet overcomes many of their shortcomings.

7.1 Introduction

Our approach is a semi-automatic method because it needs the user to identify the VO of interest in the first frame. Then the algorithm will locate and extract the object with pixel-wise accuracy from every subsequent frame automatically. The typical technique that have been adopted in semi-automatic approaches is *representation and localization*. To effectively represent the object, a variety of models has been proposed including: 2-D mesh [26] [27], binary model [28], color histogram [89], deformable
Then the model will be placed to all the possible positions in subsequent frames and the object will be localized where the best match is found. In order to measure the quality of the match between the model and object candidates, a similarity function needs to be defined, which traditionally considers only the information of the object such as spatial similarity and temporal consistency.

The importance of integrating background information in the matching process is demonstrated in [89]. More specifically, it takes into account the dissimilarity between the object and the background by down-weighting the colors that appear in both classes in the similarity function such that the object is represented only by the salient parts. Avidan extends this idea by explicitly treating tracking as a classification problem [34]. Single object tracking, for example, requires identifying each pixel as object or background, and therefore can be formulated as a binary classification problem. One limitation of Avidan’s work is that it only provides the bounding-box as the results. The same classification-based spirit appears in [32] and our earlier work [33] for the task of VO extraction where the tracking results are required to be pixel-wise accurate.

In spite of the differences in detailed algorithms, these classification-based approaches can be abstracted as a generic four-step model: (1) construction of feature vectors; (2) training of classifiers; (3) classifications applied to new frames; and (4) object generation. The first step is to design a feature representation for every pixel. It may be the raw chromatic values such as RGB [32] or the histogram of colors [34]. In our approach, we extract so-called local and neighboring features from block regions around the pixels of interested such that they are not only represented by the
chrominance or luminance values but also some of the spatial structures among them. In the second step a classifier is trained and the classification function is obtained to discriminate the pixels that belong to the object from those belong to the background. Different classifiers have been attempted such as neural networks [32] and an ensemble of linear classifiers [34]. To achieve high classification accuracy, we propose multi-layer classification which adopts SVM, Soft SVM, and \( \psi \)-learning as the baseline classifier. The third step is to evaluate the classification function at pixels in subsequent frames. Finally, object is segmented based on the classification results for which the way of implementation varies. For example in [34] a so-called confidence map is first produced according to the classification results, and tracking is then realized by locating the object where the peak of the confidence map occurs. The output of the tracker, however, is a rectangle that tightly encloses the object of interest. For the task of VO extraction the fourth step can even be skipped [32] since after the classification step we already know for every pixel if it belongs to the object. However, for efficiency purpose it is not necessary to do the classification pixel by pixel. By exploiting the spatial redundancy, we introduce the block-level classification instead and design a pyramid refining scheme to refine the boundary in an efficient and scalable manner.

Starting from the next section we will illustrate in details how the proposed approach works for single VO extraction. The extension to multiple VO scenario will be presented in Section 7.7, which is followed by experimental results.
7.2 Approach Overview

Fig. 7.1 presents an overview of our approach for single VO extraction. As one can see, it consists of two phases: 1) the training phase, and 2) the tracking phase. The training phase begins with dividing the first frame, chosen as the training frame, into blocks that are defined as object blocks or background blocks depending on which class the pixels in the block center belong to. Every centering pixel as well as every block is represented by the local and neighboring features. Then through the multi-layer learning, a set of linear decision functions that are stored in a tree structure are obtained. In the tracking phase, each subsequent frame is also divided into blocks, and for each block the set of decision functions are evaluated to decide whether the pixel at its center belongs to the object or not, which consequently determines the class label of the block. Finally, the tracking mask is formed by all the identified object blocks. At this point the resolution of object’s boundary is as large as the size of the block. To obtain pixel-wise accuracy, we design a so-called pyramid boundary
refining algorithm which is able to refine the object boundary in an efficient and scalable manner.

Comparing with previous works, our method has following advantages:

1. **Low computational complexity.** The time-consuming processes of object modeling, segmenting, and searching are avoided. Instead, VO extraction is achieved through the testing phase of the learning machine, which only requires evaluation of a small number of linear functions. Furthermore, classifying pixels at the block level instead of pixel level saves even more computations. As a result, our approach has low computational complexity;

2. **Robust to motion fluctuation.** No motion assumption has been imposed, which makes the approach robust to motion fluctuation. As will be demonstrated by the experimental results, it can perform well even when the object stays still for arbitrarily long period of time or when its different parts exhibit different motion characteristics;

3. **Robust to occlusion.** Occlusion is a very challenging scenario for both automatic and semiautomatic approaches. The proposed approach, in contrast, by decomposing object into blocks, is able to recognize un-occluded object portions as long as they still exhibit the object features.

In the following four sections, the major four components of the approach, namely feature extraction, multi-layer learning and classification, block-level recognition, and pyramid boundary refining, will be explained sequentially.
Figure 7.2: The similarity between object (human body) and the background in Silent sequence. (a) Some background areas (in circles and ellipses) are similar to the woman’s face. (b) Some background areas (in circles and ellipses) are similar to the woman’s hair.

7.3 Feature Extraction

VO extraction requires the object to be tracked at the pixel-wise resolution. If we represent individual pixels utilizing pixel-wise color or intensity information as most approaches do, a lot of misclassifications would occur due to the negligence of the support of the spatial relationship among pixels. Take the silent sequence as an example, assuming that the human body is the object of interest. As shown in Fig. 7.2, the background contains a large amount of small areas whose chrominance characteristics are very close to those of the face or hair regions. As a result, many background pixels will be tracked as those of the human body. To confront this problem we extract features from block regions centering at the pixel of interest, which describes a pixel not only by its chrominance or luminance values, but also
some of the spatial structures among them. In this way more reliable classifications is able to be rendered.

Associated with each pixel there are two types of blocks defined for the feature extraction purpose: unit blocks and neighboring blocks. A unit block is the smallest block we are dealing with in our algorithm. More specifically, it has the size of $9 \times 9$ pixels and its centering pixel is what we want to represent and classify. Neighboring blocks, as the name suggests, are the 8-connected neighbors of the unit block as shown in Fig. 7.3. Two types of features are constructed accordingly: local features, denoted as $\vec{f}_{local}$, and neighboring features $\vec{f}_{neighbor}$, which are defined as follows.

### 7.3.1 Local Features

Local features extraction procedure collects the information from a unit block by applying the Discrete Cosine Transform (DCT) and constructing a feature vector as follows:

$$
\vec{f}_{local} = (f_0, f_1, f_2, f_3)^T = \left( \begin{array}{c} c(0, 0) \\ \sqrt{\sum_{j=1}^{N-1} c(0, j)^2} \\ \sqrt{\sum_{i=1}^{N-1} c(i, 0)^2} \\ \sqrt{\sum_{i=1}^{N-1} \sum_{j=1}^{N-1} c(i, j)^2} \end{array} \right),
$$

(7.1)
where \( c(i,j) \) are the DCT coefficients, \( f_0 \) is the average intensity, and \( f_1 \) and \( f_2 \) represent the horizontal and vertical edges, respectively. All the other high frequency information is contained in the last component \( f_3 \). Because of the unbalanced energy distribution among coefficients, many high frequency components \( c(i,j) \) are close to zero. For this reason, we set \( N = 3 \) in Eq. (7.1) and use only the first 9 DCT coefficients when to calculate \( \vec{f}_{local} \).

### 7.3.2 Neighboring Features

In contrast to unit blocks, neighboring blocks contribute to the extraction of neighboring features. For a \( 9 \times 9 \) unit block \( B_0 \), its neighbors are eight \( 9 \times 9 \) blocks that are adjacent in the vertical, horizontal and diagonal directions. With \( \text{avg}(B_i) \) denoting the average intensity of block \( B_i \) we compute the neighboring features as:

\[
\vec{f}_{neighbor} = \left( \begin{array}{c} \text{avg}(B_1 + B_2 + B_3) \\ \text{avg}(B_3 + B_4 + B_5) \\ \text{avg}(B_5 + B_6 + B_7) \\ \text{avg}(B_7 + B_8 + B_1) \end{array} \right).
\]

The calculations given above only consider the grayscale information. When the video sequence is chromatic, we compute Eq. (7.1) and Eq. (7.2) for each color component and then concatenate the vectors respectively to form the chromatically local and neighboring features.

The purpose of introducing both local and neighboring features is to make classification efficient and effective. The 4-dimensional (or 12-dimensional for chromatic sequences) local feature, rather than all DCT coefficients, reduces the data amount for representation while the neighboring features help separate the pixels that are similar when only local features are considered. Those two features will be used in the multi-layer learning and classification process.
7.4 Multi-Layer Learning and Classification

Linear classification yields good performance when the object can be easily separated from the background. When the object and the background become complex and share some common features, the classification boundary tends to be nonlinear. Fig. 7.5(a) gives such an example, in which one linear decision function does not completely separate the object samples from the background. Consequently, a significant portion of the object is identified as background or vice versa. Some nonlinear decision functions have been investigated to solve this problem which unfortunately impose a high computational cost and remain to be an open topic of study technically [90].

We propose a hierarchical partition scheme which breaks the initial training set into many subsets, each of which contains samples that are more likely separated by a linear boundary. In other words, piecewise linear hyperplanes are used to approximate
the nonlinear boundaries. Previously, [91] and [92] used this idea to yield better classification performance and to reduce the computation time.

We further propose a multi-layer method that partitions the training set sequentially according to the results of the previous classification step. Instead of only one classifier, this method yields a hyperplane decision tree consisting of all the hyperplanes that are used to divide the training set. Each node of the tree represents one hyperplane, denoted as $HP_s^l$ where the superscript $l$ represents the level of the node while the subscript $s$ denotes the path from the root to the current node, as shown in Fig. 7.4(a). After the first separation, each subset may still contain both object and background samples. Two hyperplanes are then generated to separate the two subsets, respectively. In Fig. 7.4(a), we use $R$ and $L$ to represent the two hyperplanes.

The hyperplanes along a path from the root to any one leaf node will eventually separate the object from the background. Fig. 7.5(c) displays two linear boundaries which are obtained when our approach is applied to the same samples in Fig. 7.5(a), while the constructed hyperplane tree is depicted in Fig. 7.4(b). For this particular case, the hyperplane tree is unbalanced because one subset after the first separation by $HP^1$ has only the object samples so that no more separation is needed. On the other side of $HP^1$ the subset has both object and background samples, which are further separated by $HP^2_R$.

It is important to note that $\psi$-learning is more suitable than SVM for this multi-layer approach. As discussed in Chapter 2, due to the shape of the cost function, $\psi$-learning is more robust against the misclassified samples while SVM is more sensitive. Thus the hyperplane is aligned more closely to the local boundaries of the two classes of samples by $\psi$-learning than by SVM. The hyperplanes generated by SVM
Figure 7.5: Illustration of the multi-layer method. (a) 100 training samples $S_i$ with coordinates $(x_1^i, x_2^i) = (\rho_i \cos \theta_i, \rho_i \sin \theta_i)$ are randomly generated in the left hand side circle of the unit disk. The sample $S_i$ is labeled as “background” if $\theta_i \in \left[\frac{2\pi}{3}, \frac{4\pi}{3}\right]$. Otherwise, it is labeled as “object”. (b) A hyperplane $(w = [28.1, 32.2]^T, b = -4.3)$ is obtained by training the first layer. (c) One additional hyperplane $HP^2_R$ $(w = [29.2, -22.8]^T, b = -1.5)$ is obtained by training the second layer. (d) A different hyperplane $(w = [5.4, 0.27]^T, b = 0.9)$ is obtained when the first layer is trained using SVM.
are strongly influenced by the global distribution of the training samples which is contrary to the objective of the multi-layer approach. For example, the hyperplane $HP_1$ obtained by SVM for the cluster of Fig. 7.5(a) is shown in Fig. 7.5(d). Evidently, the SVM approach generates a compromise for all the samples and thus not suitable for further separation of misclassified blocks.

Technically, the multi-layer scheme takes the following two steps. The first step is to generate the hyperplane tree. It begins with the initial training set $S = B \cup O$, where $B$ and $O$ represent the set of the background and the object, respectively. By training the learning machine using all the samples in $S$, the first hyperplane $HP^1$ representing the root of the tree is obtained. Depending on which side of $HP^1$ they are on, the samples in $S$ is partitioned into two subsets denoted as $\hat{B}$ and $\hat{O}$. Usually, $\hat{B} \neq B$ and $\hat{O} \neq O$ because there always exist some background samples that are wrongly classified as the object by $HP_1$ and vice versa. If so, $\hat{B}$ and $\hat{O}$ are trained independently to obtain two additional hyperplanes, denoted as $HP^{2}_{R}$ and $HP^{2}_{L}$, respectively, and the tree size grows to two levels. At this point, the training set is divided into four subsets. If necessary the four subsets will be partitioned again, so forth and so on. In general, the more levels the tree has, the smaller the subsets which $S$ is broken into. This process continues until the percentage of the misclassified samples in all the new subsets is no greater than a predetermined threshold $\epsilon$, which is set to be 0.05 throughout our experiments.

Once the hyperplane is obtained using the approach just described, it can be used to classify the pixels $(i, j)$ in the subsequent video frames. It follows a sequential classification procedure starting from the root of the tree and ending at a leaf node. Every time a node is encountered, the corresponding decision function is evaluated.
For an intermediate node, the sign of the result determines which branch of the tree to go: positive sign directs to the left and negative the right, for example. Finally, at a leaf node the sign indicates the class: object or background, and the class label of the pixel which is denoted as $C_\psi(i, j)$ is accordingly obtained: 1 or -1.

Now with the local features, the neighboring features, and the multi-layer $\psi$-learning tool, we are ready to extract the object. The most straightforward method is to calculate $C_\psi(i, j)$ for every pixel and then conform the object by all the pixels whose class labels are 1. For the video sequences we experiment with, the maximum number of layers yielded by the multi-layer method is three when we choose $\epsilon = 0.05$. So in order to determine the class label of one pixel we just have to evaluate no more than 3 linear functions, which evidently requires low computational complexity. Two examples of the tracking mask after different layers are given in Fig. 7.6 and Fig. 7.7, respectively.

### 7.5 Classification at the Block Level

It has been shown at the end of the previous section, we can achieve VO extraction through pixel-by-pixel classification. Yet in most video frames there is abundant spatial redundancy that we can take advantage of to make the tracking step more efficient. Let $p$ denote a pixel and $N(p, d)$ the set of pixels within a small distance $d$ from $p$. Due to the spatial redundancy of images, the class labels of $p$ and $N(p, d)$ tend to be consistent with each other. In other words, if $p$ belongs to the object then it is very likely that $N(p, d)$ belong to the object too, except for the pixels lying around the object boundary. Based on this observation, we introduce the concepts
of object blocks and background blocks, and suggest the classification be done at the block level.

Defining an object block as a block whose centering pixel belongs to the object and an background block otherwise, we propose a block-level classification method which is summarized as follows:

1. divide current frame into blocks of size \((2^N + 1) \times (2^N + 1)\) with one pixel overlapping in both vertical and horizontal directions;
2. calculate $\vec{f}_{local}$ and $\vec{f}_{neighboring}$ of the centering pixel;

3. evaluate the set of decision functions that have been trained through multi-layer $\psi$-learning to determine the class labels of the centering pixels as well as the labels of blocks;

4. classify all the pixels within the block as object if the block is an object block; otherwise as background.

It would be more common if we had used the block size $2^{N_0} \times 2^{N_0}$. However, an odd number of pixels is preferred in our approach because of the necessity of “centering pixel”. For this reason, the block size is chosen to be $(2^{N_0} + 1)$ pixels in width and height. As for the introduction of one pixel overlapping, we will give the reason in the next section.

Significant saving in the computational cost is one of the benefits of this block-level representation and classification method. By using the block level classification as explained above, an image of size $M \times N$ is decomposed into exact $L_M \times L_N$ blocks. In other words, we have $M = 2^{N_0}L_M + 1$ and $N = 2^{N_0}L_N + 1$ for some integers $L_M$ and $L_N$. 

Figure 7.8: The extracted VO of Mom sequence using block-level classification when block size is $9 \times 9$. (a) Frame 1. (b) Frame 118. (c) Frame 138.
For that decomposition, the DCT and multi-layer classification is computed by \( L_M \times L_N \) times instead of \( M \times N \) times because we only have to compute \( C_\psi(i,j) \) for the centering pixel of each block. Therefore the computation is theoretically reduced by

\[
\gamma = 1 - \frac{L_M L_N}{MN} = 1 - \frac{M-1}{2^{N_0}} \frac{N-1}{2^{N_0}} \geq \frac{2^{2N_0} - 1}{2^{2N_0}}
\]  

(7.3)

in comparison with pixel-by-pixel classification, which converges to 1 quickly with the increase of \( N_0 \).

The image size is an important factor to consider when we choose the value of \( N_0 \), and usually large images can have relatively large \( N_0 \). In the meantime, the size of the object should also be taken into consideration. If the object we intend to track is quite small, a big block size will not be appropriate. Through the experiments, we find \( N_0 = 3 \) (the block size is \( 9 \times 9 \)) is a good choice for the video sequences we are working with, and in that case the computation reduction would be around \( \frac{63}{64} \approx 98.4\% \) according to Eq. (7.3).

As shown in Fig. 7.8, the block-level classification scheme is quite effective even when the object undergoes considerable deformation. The drawback, however, is the stair-like object boundary due to the block effect. Although this coarseness is tolerable in some applications such as target positioning, many others do require pixel-wise accuracy. To address this problem, we propose a pyramid boundary refining algorithm which refines the object boundary in an efficient and scalable way and will be explained in the next section.
7.6 The Pyramid Boundary Refining Algorithm

Fundamentally the pyramid boundary refining algorithm is an iterative process that keeps refining the object boundary until the pixel-wise resolution is reached. During the refining process, a class map $CM^L$ is maintained as a binary image that stores the segmentation result obtained after $L$ iterations. The pixel value of $CM^L$ is defined as:

$$CM^L(i, j) = \begin{cases} 
-1, & \text{if pixel } p(i, j) \text{ is identified as background after } L \text{ iterations;} \\
1, & \text{otherwise.}
\end{cases}$$

Some regions in the class map $CM^L$ are identified as the boundary zone ($BZ^L$) in which the boundary is possibly located and therefore the pixels’ class assignments present ambiguity. Initially $BZ^0$ is a quite large area. In order to reduce the uncertainty about the boundary’s actual location, a special group of pixels in the boundary zone, which are named as boundary seeds (BSs) and denoted as $S^0_{BS}$, are selected for class label checking. In other words, their $C_p(i, j)$ are computed. According to the newly obtained class labels, the class map is updated such that the block size around the boundary is decreased. Also $BZ^0$ is reduced to $BZ^1$ which is only half as large. The similar process continues to increase the boundary resolution until no refinement is needed.

In the following two subsections, we will explain the initialization and iteration steps of the pyramid boundary refining algorithm in detail.

7.6.1 Initialization Step

The algorithm starts with the block-level classification discussed in Section 7.5. By dividing the frame into blocks of size $(2^{N_0} + 1) \times (2^{N_0} + 1)$, we obtain the initial
segmentation result $CM^0$ as follows:

$$CM^0(i, j) = C_\psi(i_n, j_n), \ |i - i_n| \leq 2^{N_0-1}, \ |j - j_n| \leq 2^{N_0-1}$$

(7.5)

for all $p(i_n, j_n) \in S_c$, where $S_c$ is the collection of the centering pixels of all blocks.

The next step in initialization is to determine the boundary zone $BZ^0$. Based on the assumption that the boundary is within the regions that exhibit transitions between object and background blocks, we first identify the transition areas in $CM^0$ that are defined as the union of the transition blocks which have at least one eight-connected neighboring block belonging to a different class. Then the transition areas are decomposed into so-called transition tiles (TTs), which are rectangular regions containing $2 \times 2$ transition blocks. Suppose at this initialization step the block size is $9 \times 9$ ($N_0 = 3$), and so the transition tiles are of size $17 \times 17$. Some transition tiles in $CM^0$ are shown in Fig. 7.9, in which the pixel is portrayed as “o” if it takes value 1, and “+” otherwise. An interesting phenomenon about the transition tiles is that if we consider “+” and “o” as symmetric and further ignore the tiles’
orientation there are actually only three distinct transition patterns. As shown in Fig. 7.9, these three patterns convey different boundary information and thus need to be handled differently. The occurrence of pattern #1 implies a steep or nearly vertically located boundary in the tile, and therefore a rectangular boundary zone is designed as depicted in Fig. 7.10(a). If the slope of the boundary is relatively moderate, we get pattern #2 and accordingly the boundary zone is conceived as a “L” shape (Fig. 7.10(b)). As for pattern #3, the boundary is assumed to be in the middle and form a cross shape (Fig. 7.10(c)). By combining the boundary zone in all transition tiles of $CM^0$, $BZ^0$ is obtained.

### 7.6.2 Iteration Step

Fig. 7.11 gives a diagram of the core operations of the iteration step of the proposed refining algorithm. To explain the iteration step more clearly, we use $N_0 = 3$ as an example to show how the algorithm works during the first iteration before giving the general updating equations for $CM^L$ and $BZ^L$. 

![Figure 7.10: The boundary zones (the blank areas) and boundary seeds (△) determined for different patterns.](image)
After the determination of $CM^0$ and $BZ^0$, we have roughly known where the object boundary is. Its actual location, however, is still uncertain. In order to reduce the uncertainty, the pixels that lie in the middle of the 2-D boundary zone, which are depicted as $\triangle$ in Fig. 7.10, are selected as boundary seeds $S^1_{BS}$ for class label checking. More specifically, $S^1_{BS}$ is constructed as the following:

$$
S^1_{BS} = \{p(i, j) \mid i = 4m + 1, j = 4n + 1, \text{ and pixel } p(i, j) \text{ falls inside } BZ^0\}, \quad (7.6)
$$

where $m$ and $n$ are positive integers.

Then for each element $s_n(i_n, j_n) \in S^1_{BS}$, we apply the multi-layer classification and determine its class label $C_\psi(i_n, j_n)$, with which $CM^1$ and $BZ^1$ can be generated.

1. $CM^0 \rightarrow CM^1$
The class map $CM^1$ is updated from $CM^0$ as the following:

$$CM^1(i, j) = \begin{cases} 
C_\psi(i_n, j_n), & \text{if } \exists s_n(i_n, j_n) \in S^1_{BS} \text{ such that } |i - i_n| \leq 2, |j - j_n| \leq 2; \\
CM^1(i, j), & \text{otherwise.}
\end{cases} \quad (7.7)$$

The updating operation of the class map has two important properties. First, it does not affect the pixels falling outside of the boundary zone. As a result the segmentation results only experience small changes around the object boundary area. Secondly, the class labels of the pixels within the boundary zone are updated again at the block level as indicated in Eq. (7.7), hence the extracted object would still has the stair-like boundary at this point. However the block size, which is $5 \times 5$ now, is smaller than that of the initialization step. As a result, the block effect shown near the boundary has been reduced as one can see from the tracking mask shown in Fig. 7.14(c).

2. $BZ^0 \rightarrow BZ^1$
Figure 7.13: Three transition tiles in $CM^1$ ($N_0 = 3$). (a) The transition tile of pattern #1. (b) The transition tile of pattern #2. (c) The transition tile of pattern #3.

Fig. 7.12 provides an example which considers a transition tiles of $CM^0$ that shows the pattern #1 to illustrate how the boundary zone can be further reduced according to the newly updated class map $CM^1$. Suppose the class labels of its boundary seeds are identified as shown in Fig. 7.12(a) and the class map is updated accordingly (Fig. 7.12(b)). Although the boundary searching strategy remains the same, which is to focus on the areas showing the transition between the object and background, the size of the transition tiles becomes smaller. Three new transition tiles, each of which contains only $9 \times 9$ pixels now, are highlighted in Fig. 7.13. In spite of the smaller range, these transition tiles fortunately manifest very similar patterns as discussed in Fig. 7.10, and therefore their boundary zones and boundary seeds can be determined in a similar way. The union of the boundary zones in all transition tiles of $CM^1$ constitute the $BZ^1$, which are shown in Fig. 7.12(c). As one can see, the area of $BZ^1$ is nearly half as large as $BZ^0$ and with it the uncertainty about the actual boundary location is reduced.
Now, with the new class map $CM^1$ and new boundary zone $BZ^1$ available, we are ready for the next iteration which will go through the same steps as explained above. In general, the updating equations for each iteration can be summarized as the following.

- updating $S_{BS}^L$

$$S_{BS}^L = \{p(i,j)|i = 2^{N_0-L}m+1, j = 2^{N_0-L}n+1, \text{ and pixel } p(i,j) \text{ falls inside } BZ^{L-1}\},$$  

(7.8)

where $m$ and $n$ are positive integers, and $N_0 = 3$ for $9 \times 9$ initial block size.

- updating $CM^L$

$$CM^L(i,j) = \begin{cases} 
C_\psi(i_n,j_n), & \text{if } \exists s_n(i_n,j_n) \in S_{BS}^{L-1} \text{ such that } |i - i_n| \leq \lfloor \frac{S_L}{2} \rfloor, |j - j_n| \leq \lfloor \frac{S_L}{2} \rfloor; \\
CM^L(i,j), & \text{otherwise},
\end{cases}$$  

(7.9)

where $S_L = 2^{N_0-L}$.

- updating $BZ^L$

$$BZ^L = \bigcup_i BZ(TT_i^L),$$  

(7.10)

where $TT_i^L$ is the $(2^{N_0-L+1} + 1) \times (2^{N_0-L+1} + 1)$ transition tiles determined in $CM^L$, and $BZ(TT_i^L)$ denotes the boundary zone in the tile $TT_i^L$.

As one can see from the updating equations, the larger the $L$, the smaller the boundary zone $BZ^L$. When $L = N_0$ the boundary zone is only one pixel width and the $C_\psi(i,j)$ of the boundary seeds does not effect other pixels any more, which means the pixel-wise resolution is reached and therefore the iteration process stops. Fig. 7.14 shows the segmentation results of the same frame but of different boundary resolutions. The pyramid boundary refining algorithm works so effectively that almost the
same tracking results are observed in Fig. 7.14(e) and Fig. 7.14(f), which are obtained by the proposed refining algorithm and pixel-by-pixel classification respectively.

It is self-evident that the block effect around the object boundary is eliminated at the expense of the increased computational complexity. As a result the processing speed is surely not as fast as the block-level classification. However because the multi-layer classification is carried out only on the pixels selected as the boundary seeds, the run time is reduced to about \( \frac{1}{10} \) of that of the pixel-by-pixel classification.
method. Another important property of the proposed refining algorithm is its flexibility. Depending on different applications, the iteration process can stop whenever the desired boundary resolution is reached. Hence by our approach the object boundary is able to be refined in an efficient and scalable manner.

It is worthy pointing out that because of the important role played by the boundary seeds in the proposed refining algorithm, we need to guarantee the integer coordinates for them such that they are available as image pixels. This is the reason why we introduce the one pixel overlapping between the classification blocks in Section 7.5.

7.7 Extension to Multiple VO Extraction

The extension to multiple VO extraction is straightforward. In analogy to binary classification, an $M$ object extraction problem can be formulated as an $(M + 1)$-category classification problem. That is, one class for the background and $M$ classes for the objects of interest. Most of the mechanisms presented above, such as block classification and pyramid boundary refining, are still applicable. The only change one needs to make is to replace the binary classifier with a multi-class classifier. Conceptually it is the case. Unfortunately, as far as the implementation is concerned, the extension is far more difficult than it appears because multi-category classification is still an ongoing and immature research topic itself in machine learning, which explains why most of the results reported by classification-based approaches are limited to single object scenarios. Only recently have works emerged to offer new tools such as multi-category $\psi$-learning that can help tackle the multi-object problem. Here we presents an attempt of such. As pointed out in Chapter 2, $\psi$-learning aims at the
minimization of GE and therefore has the advantages in nonseparable cases. For this reason, we employ binary $\psi$-learning as the classifier in [33]. To tackle the challenging task of multi-object extraction, multi-category $\psi$-learning, which retains the desirable properties of its binary counterpart, has to be employed.

The pipeline is shown in Fig. 7.15. Suppose we have $M$ VOIs of interest. Similar to single VO extraction, the training phase begins with dividing the first frame, chosen as the training frame, into $(M+1)$ types of blocks (the number of different VOIs plus background) depending on which object or background the pixel at the center of the block belongs to. As before, the block size is empirically chosen as $9 \times 9$. The same method of feature extraction described in previous sections are used to represent each block as well as the centering pixels. Now with the training data in place, the next step is to train the machine by solving the optimization problem Eq. (2.15),

Figure 7.15: An overview of the proposed approach for multiple VO extraction.
which yields \((M+1)\) decision functions that separate the \(M\) objects as well as the background.

In the tracking phase each subsequent frame is also divided into blocks of 9×9, and for each block the \(M+1\) decision functions are evaluated to decide what object the centering pixel belongs to, which consequently determines the class label of the block. Then the tracking mask of every object is formed by the blocks that have been classified in the corresponding class, at which point the resolution of object’s boundary is as large as the size of the block. Then after the pyramid boundary refining is applied, the object boundary can be refined and the pixel-wise accuracy can be achieved.

7.8 Experimental Results

To test the effectiveness and robustness of the proposed approach, we apply it to several standard MPEG-4 test video sequences, which exhibit certain varieties of temporal and spatial characteristics. These sequences are *Akiyo, Mom, Mom & Daughter, Silent, Flower Garden, Students*, and *Trevor*. All experiments are carried out on a Pentium IV 2.5-GHz PC. The performance is evaluated on both subjective and objective basis, and the computational efficiency of the approach is also discussed.

7.8.1 Single Object Extraction Using \(\psi\)-Learning

We start with the scenario of single VO extraction, and \(\psi\)-learning is employed as the classifier for its advantages over SVM for the case of nonseparable data as well as multi-layer learning which are discussed in Chapter 2 and Section 7.4 respectively.
Subjective Results

Akiyo and Mom belong to the typical head-and-should type of sequences. The objects which are the anchored-women in the scene exhibit slow and smooth motion against a stationary background. The performance of our approach is satisfactory even when the objects undergo considerable deformation, as shown in Fig. 7.16 and Fig. 7.17.
Mom & Daughter is another typical head-and-should type of sequence. However, it exhibits much more complex motion characteristics than Akiyo and Mom. If mom and daughter are considered as a single object, we have to deal with its non-coherent motions: the mom’s head and shoulder move slowly, the daughter stays nearly still for most of the time, and the mom’s left hand even disappears in the middle of the sequence. Nevertheless the proposed approach performs well too, as shown in Fig. 7.18.

The third test sequence is Silent, in which a woman makes a number of different gestures. If Mom & Daughter is characterized as the combination of “slow motion (mom) along with still motion (daughter) over a simple background”, the Silent can
be considered as the combination of “rapid motion (woman’s hands) and slow motion (woman’s body) over a textured background”. The extracted VOs from several frames are provided in Fig. 7.19, showing the effectiveness of our approach.

Among the four sequences tested in the experiments, *Sun Flower Garden* is perhaps the most challenging one. Unlike the previous video-conference kind of sequences, it displays a natural scene that is rich of colors and textures with a non-stationary camera. In addition, the selected VO, which is the houses, is only partially viewable for quite a few frames. The presence of occlusion adds another difficulty to this sequence. Other approaches such as template matching and motion tracking
may fail in this case. In contrast, our approach can survive this problem because the un-occluded portion that exhibits the features of the object is still recognizable by our approach. The results of the Flower Garden sequence, shown in Fig. 7.30, demonstrate this advantage. As we can see from Fig. 7.30, some portions of the houses are uncovered and extracted correctly as the camera moves along. At the same time, the newly occluded area is identified as background and does not appear in the extracted regions. When the occlusion finally disappears, the entire houses emerge as a complete object.

It can also be observed that even when the camera is in motion and the training is only done once, the tracking results are still of good quality. We believe this is
because there is no significant change of the video content in the *Flower Garden* so that the information captured by the first frame is rich enough to generate a classifier that is robust for the rest of the sequence. Otherwise, a retraining may be necessary. To do so, a scene change module should be incorporated into the system to detect the change of the video content, which can be measured by the difference of the color or texture histogram between frames, and signal the necessity of the retraining when the difference is significant.

**Objective Results**

So far, the proposed approach is evaluated on a subjective basis. In recent year, a number of measures have been proposed to objectively assess the performance of video segmentation and tracking [93][94]. When the ground-truth segmentation maps are available, which is the case for the sequences of *Akiyo, Mom & Daughter* and *Sun Flower Garden*, the so-call relative quality evaluations are applied. As the name suggests, relative evaluations measure the similarity between the reference segmentation and the estimate segmentation obtained by certain VO extraction methods. The typically employed metrics include shape fidelity [93], the similarity of spatial features such as areas [95], the consistence of the trajectories and velocities of the objects [96], or a combination of both spatial and temporal distortions [97]. Here we adopt the criterion proposed in [98]. Let $VO_n^{\text{est}}$ and $VO_n^{\text{ref}}$ denote the estimated and the reference binary object mask of frame $n$. Then according to [98], the spatial distortion of $VO_n^{\text{est}}$ is defined as:

$$d(VO_n^{\text{est}}, VO_n^{\text{ref}}) = \frac{\sum_{(x,y)} VO_n^{\text{est}}(x, y) \oplus VO_n^{\text{ref}}(x, y)}{\sum_{(x,y)} VO_n^{\text{ref}}(x, y)}, \quad (7.11)$$
where the $\oplus$ is the binary XOR operation. This measure is simple to compute and has been widely used [20][32]. Note that the numerator is equal to the number of wrongly classified pixels while the denominator is the number of pixels per frame. So fundamentally Eq. (7.11) is a measurement of the classification error which makes it very suitable to evaluate our approach.

Fig. 7.21 shows the segmentation error of the Mom & Daughter, Akiyo, and Sun Flower Garden sequences. Also the error rates using SVM as the classifier are provided for the comparison purpose. Evidently $\psi$-learning outperforms SVM for the proposed approach.

**Average Run Time**

The average run time of the approach is shown in Table 7.1. Comparing with the pixel-by-pixel classification method whose run time is around 4.65 seconds for a 176×144 frame and 3.969 seconds for a 180×120 frame, the proposed method is about ten times faster. Run time analysis shows that the feature extraction operation takes nearly 99.7% of the whole run time, and DCT is the major contributor. While the implementation of the algorithm can be further optimized, it should be mentioned that so far we have only considered the intra-frame information for segmentation. The run time is expected to be reduced significantly when the temporal redundancy

<table>
<thead>
<tr>
<th></th>
<th>Akiyo</th>
<th>Mom</th>
<th>Mom &amp; Daughter</th>
<th>Silent</th>
<th>Flower Garden</th>
</tr>
</thead>
<tbody>
<tr>
<td>average (msec)</td>
<td>396.1</td>
<td>332.5</td>
<td>378.9</td>
<td>413.4</td>
<td>439.3</td>
</tr>
<tr>
<td>std (msec)</td>
<td>0.008</td>
<td>0.013</td>
<td>0.014</td>
<td>0.022</td>
<td>0.030</td>
</tr>
<tr>
<td>frame size (pixels)</td>
<td>176×144</td>
<td>180×120</td>
<td>180×120</td>
<td>176×144</td>
<td>176×144</td>
</tr>
</tbody>
</table>

Table 7.1: Average and standard deviation of run time per frame using $\psi$-learning.
Figure 7.21: Segmentation error with respect to frame numbers. (a) Akiyo Sequence. (b) Mom & Daughter sequence. (c) Flower Garden Sequence.
is utilized, by which one can reduce the number of pixels whose class labels have to be obtained through feature extraction and the multi-layer procedures. The potential of that reduction will be discussed later in the chapter of conclusions.

### 7.8.2 Single Object Extraction Using Soft SVM

Recall that in the training phase the training frame is divided into blocks and each of them is labeled depending on which class the centering pixels belong to. As mentioned before, we observe during the experiments that there are uncertainties or ambiguities about the class labels of the blocks that lay around the object boundary, i.e., they can not be fully assigned to either one of the two classes. Doulamis et. al. [32] consider these boundary-adjacent pixels to be in a region of uncertainty and exclude them from the training set so as not to confuse the training of the classifier. Unfortunately, the boundary-adjacent pixels play a critical role in defining accurate boundaries of VOs. Without the latter, the performance of any content-based video processing algorithms relying on VO extraction will be fundamentally affected. We argue that the boundary-adjacent pixels carry useful information regarding the transition from one class to the other which should be included in the training process to achieve an accurate boundary. For this reason, we conduct a series of experiments that employs S_SVM as the classifier.

The question that needs to be answered first is how to generate the real-valued membership $y_i \in [-1, 1]$ for a given block $B_i$. For this specific application, we adopt the method described in Eq. (3.14) as the following. First, we use the normalized number of object and background pixels contained in the block $B_i$ to define the
conditional probability. More specifically, for the block size of $L \times M$, we have

$$\begin{cases} 
P(Y_i = 1|X_i = B_i) = \frac{\# \text{ of object pixels}}{L \times M}, \\
P(Y_i = -1|X_i = B_i) = \frac{\# \text{ of background pixels}}{L \times M}. 
\end{cases}$$  \tag{7.12}$$

Then according the Eq. (3.14), the soft membership $y_i$ is obtained as

$$y_i = \frac{\# \text{ of object pixels} - \# \text{ of background pixels}}{L \times M}. \tag{7.13}$$

Obviously, $y_i$ is the normalized difference between the number of object and background pixels contained in the block. When the block locates completely inside (or outside) the object, we have $y_i = 1$ (or $y_i = -1$) showing no labeling ambiguity at all. When the block contains equal number of object and background pixels, $y_i = 0$ which indicates the maximal uncertainty. In other cases, $y_i$ varies between -1 and 1.

**Subjective Results**

The major thesis of this series of experiments is to demonstrate that by factoring the information of certainty $S_{SVM}$ can achieve higher classification accuracy than SVM. To serve this purpose, the performance comparison will be focused on SVM, $S_{SVM}$ and FSVM proposed in [42].

Some original frames of *Akiyo* and the extracted VO by $S_{SVM}$ as well as SVM are given in Fig. 7.22, which show that the objects extracted by $S_{SVM}$ are more complete than that by SVM. The errors concentrate on the area of the lady’s dark hair where the transition from the object to the dark background is blurring and consequently the blocks exhibit similarities. Evidently, $S_{SVM}$ is more accurate in classifying the “confusing” hair regions. The performance of SVM improves after the second layer classification is applied as shown in Fig. 7.23, but again it is outperformed by $S_{SVM}$ which yields smoother and more accurate contour of the object.
Figure 7.22: Applying SVM and S_SVM to Akiyo on the first layer.

Figure 7.23: The extracted VO of Akiyo after the second layer classification. (a) and (b) are from frame 4, and (c) and (d) from frame 126.
Figure 7.24: Applying SVM and S_SVM to *Mom and Daughter* using RBF kernel.

From Fig. 7.6(a), one can see that a significant part of the daughter is missing after the first-layer classification, which shows that the true boundary between the object and background classes is much more nonlinear than that of *Akiyo*. So we employ nonlinear kernels to see how nonlinear S_SVM works. Through the experiments RBF is found to be the one that yields the lowest classification errors and hence we choose RBF as the kernel function. The extracted objects by SVM and S_SVM are displayed in the second and third columns of Fig. 7.24, respectively. Obviously, the nonlinear S_SVM works well too.
We also apply S\textsubscript{SVM} to the sequence of Sun Flower Garden. Once again, RBF is employed as the kernel function and S\textsubscript{SVM} produces better results as one can see in Fig. 7.25 where S\textsubscript{SVM} recognizes the house that are cluttered by tree branches while SVM dose not.

Some different results produced by S\textsubscript{SVM}, SVM and FSVM are also displayed side by side in Fig. 7.26 which together with the previous figures, shows the superiority of S\textsubscript{SVM} over SVM and FSVM visually.
**Objective Results**

To assess the performance qualitatively and objectively, the errors yielded by SVM, S_SVM and FSVM are plotted versus the number of frames as the dashed, dotted and solid lines respectively in Fig. 7.27. Throughout the whole sequences, the solid line is below the other two lines showing that S_SVM achieves the highest classification accuracy and consequently yields the best extracted VO.

The average error of classification for each sequence is shown in Table 7.2. It should be pointed out that the absolute difference between the errors does not fully demonstrate how powerful S_SVM is considering the fact that SVM and FSVM have delivered very low classification errors already. To get an idea how much S_SVM improves with respect to SVM and FSVM, we also list the relative difference in Table 7.2, which is defined as

\[
R_1 = \frac{E_{SVM} - E_{S\_SVM}}{E_{SVM}}
\]

and

\[
R_2 = \frac{E_{FSVM} - E_{S\_SVM}}{E_{FSVM}},
\]

where \(E(·)\) denotes the classification errors yielded by the corresponding machine. The relative improvement as shown is significant. For example, for the *Akiyo* sequence, S_SVM outperforms SVM and FSVM by 31.3% and 11.0% in average, and by 51.7% and 39.7% in maximum, respectively.

It should also be noted that for the *Mom and Daughter* sequence (Fig. 7.27(b)) the dashed line and the dotted line coincide because FSVM and SVM yield the same decision function. This is not surprising because by using the RBF kernel, zero training error is attainable, and all the penalty terms \(\xi_i\) becomes zero which flattens
Figure 7.27: The comparison of the classification accuracy among SVM, FSVM and S_SVM. (a) Akiyo. (b) Mom and Daughter. (c) Sun Flower Garden.
Table 7.2: Average classification errors produced by SVM, S_SVM, and FSVM, and the relative error reduction yielded by S_SVM with respect to SVM ($R_1$) and FSVM ($R_2$).

<table>
<thead>
<tr>
<th></th>
<th>Absolute Errors</th>
<th>Relative Error Reduction</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SVM</td>
<td>S_SVM</td>
<td>FSVM</td>
<td>$R_1$</td>
</tr>
<tr>
<td></td>
<td>average</td>
<td>max</td>
<td>average</td>
<td>max</td>
</tr>
<tr>
<td>Akiyo</td>
<td>3.2%</td>
<td>2.2%</td>
<td>2.5%</td>
<td>31.3%</td>
</tr>
<tr>
<td>Mom and Daughter</td>
<td>2.4%</td>
<td>1.8%</td>
<td>2.4%</td>
<td>20.9%</td>
</tr>
<tr>
<td>Sun Flower Garden</td>
<td>4.6%</td>
<td>2.8%</td>
<td>3.1%</td>
<td>39.9%</td>
</tr>
</tbody>
</table>

Table 7.3: Comparison of average classification errors.

S_SVM is also compared against SVM when trained with only samples for which certainty measures are higher than certain thresholds. Five thresholds are tested and the average classification errors of both methods are reported in Table 7.3. For *Akiyo*, S_SVM is marginally outperformed at certain thresholds, but for the other two sequences S_SVM is significantly better. This result shows that even when SVM is trained with the samples with high certainty measures, S_SVM still performs better.
Recall that when the certainty measures all become 1, $S_SVM$ is identical to SVM. Therefore so long as there is uncertainty, $S_SVM$ is still a better choice.

**Average Run Time**

As pointed out in Chapter 3, the computational complexity for training SVM and $S_SVM$ is the same. In the tracking phase, the run time consumed by these two machines is almost identical as well. For instance, for the aforementioned three sequences the average run time per frame are 0.396, 0.388, and 0.439 seconds when SVM is applied, and 0.401, 0.395, and 0.441 seconds when $S_SVM$ is applied.

**7.8.3 Multiple Object Extraction Using Multi-Category $\psi$**

For the proposed approach to work in the multiple VO scenario, a multi-category classifier is needed to separate VOs as well as the background. Binary $\psi$-learning has shown its advantage over SVM both theoretically and experimentally. So in this subsection, we present the performance of our approach when multi-category $\psi$-learning, the multi-category classifier that retains the desirable properties of its binary counterpart, is employed.

**Subjective Results**

The first multi-VO sequence we test is *Students*. As the major content of this sequence, the two students are chosen as two objects of interest, and along with the background this is a three-class classification problem. As one can see from the original frames shown in columns (a) and (d) of Fig. 7.28, *Students* is a typical sequence of slow but heterogenous motion. For example the male student turns the head and moves his hands while his body stays still most of the time. The extracted
objects are shown in column (b), (c), (e), and (f), respectively. One can see that the proposed method works well, which discriminates the body parts of the students as well as their faces. The latter is not an easy task since the skin color is very similar between the two students.

Another sequence containing three people is also tested, and the three people are considered as three objects which makes it a four-class classification problem. The original frames and the extracted objects are shown in Fig. 7.29. Unlike the Students sequence, the objects in this sequence change the appearance a great deal. Taking the lady who sits at the farthest right as an example, her face changes from frontal to left-side view. Besides, the man in the middle is originally seated but finally standing. As seen in Fig. 7.29, the main body of the objects are successfully extracted although the boundaries of the objects are not perfectly separated due to classification errors.

The challenging video sequence Sun Flower Garden is also tested. This time, we intend to extract one more object: the tree, which makes it a two-object sequence. For the first few frames, the house is occluded by the tree. Two of such frames are shown in column (a) of Fig. 7.30, and the two extracted objects (house and tree) by using $\psi$-learning are shown in column (b) and (c), respectively. With the camera moving, the tree shifts toward the left hand side of the frame and finally disappears as in column (d). From that point on, only the house can be extracted by the proposed method as shown in the last column of Fig. 7.30.

**Objective Results**

For their simplicity and effectiveness, one-vs-all, one-vs-one and DAG are three widely-used multi-category algorithms. Suppose we have $M$ classes. As introduced in Chapter 5, one-vs-all constructs $M$ binary classifiers $f_i^{OVA}(x)$ with the $i$th one
Figure 7.28: The results of *Students*. (a) The original frames. (b) The extracted VO #1. (c) The extracted VO #2.
Figure 7.29: The results of Trevor. (a) The original frames. (b) The extracted VO #1. (c) The extracted VO #2. (d) The extracted VO #3.
Figure 7.30: The results of Sun Flower Garden. Columns (a) and (d) are original frames. Columns (b) and (e) are the extracted VO #1. Column (c) is the extracted VO #2.

separating class $i$ from all the remaining classes. One-vs-one and DAG, on the other hand, construct $M(M-1)/2$ decision functions $f_{i,j}(x)$, each of which is responsible for the binary classification task between class $i$ and $j$. At the classification step, one-vs-one classifies a sample $x$ to the class for which $f_{i}^{OVA}(x)$ produces the highest value while one-vs-one follows a voting strategy. As for DAG, it builds a directed acyclic graph using the $M(M-1)/2$ binary classifiers as the internal nodes. The classification is achieved by going through a path from the root of the graph to a leaf node which indicates the predicted class [99].

To see how multi-category $\psi$-learning performs against these three popular methods, the classification errors yielded by all the four methods are displayed every 5 frames in Fig. 7.31 where SVM is the underlying binary classifiers. For the training of each SVM, the classification accuracy is estimated by testing different values of $C$ $\in [2^{12}, 2^{11}, ..., 2^{-2}]$, and the best one is chosen for the performance comparison. As
one can see, for all the three sequences multi-category $\psi$-learning achieves the lowest classification errors almost for every test frame. Although the training is conducted only once by using the first frame, the superior generalization ability of multi-category $\psi$-learning enables it to survive nearly the whole sequence. To further see the advantage of multi-category $\psi$-learning, we replace SVM with binary $\psi$-learning in the one-vs-all, one-vs-one and DAG methods, and compare the errors in Fig. 7.32. Once again multi-category $\psi$-learning delivers the best performance.
Figure 7.32: Comparison of classification errors between multi-category $\psi$-learning, one-vs-all, one-vs-one and DAG. Binary $\psi$-learning is the underlying binary classifier employed by one-vs-all, one-vs-one and DAG.

**Computational Cost**

The computational complexity of the new approach deserves a discussion. Assume there are $M$ classes and each pixel is represented by a $d$-dimensional feature vector $x$. In the tracking phase we need to evaluate $M$ functions $f_i = w_i^T x + b_i$ each of which performs $d$ multiplications to determine the class label of a given pixel. As a result,
the computational complexity is $O(Md)$, which is a linear function of the number of objects $M$ and gives the approach low complexity and good scalability.

7.9 Summary

As the prerequisite of content-based video processing, VO extraction is an important yet challenging task. In this chapter, we present a novel semiautomatic approach that handles VO extraction as a classification problem. By this approach, we are able to overcome some limitations of conventional methods and have obtained encouraging results for various video sequences. The proposed method has the following features:

1. A multi-layer learning and classification mechanism is proposed to achieve high classification accuracy even when the sequences contain complicated content.

2. Block modeling is developed to provide a robust representation for VOs.

3. Block-level instead of pixel-level classification is designed to improve efficiency.

4. A pyramid boundary refining method is incorporated to obtain the pixel-wise object boundary in a fast and scalable manner.

In the framework of the proposed approach, another key issue is which classification technique to use. As a powerful classifier as it has been proven in a large number of applications, SVM without doubt is a good choice. The experimental results provided show that we can do even better. By using $S_SVM$ to treat the boundary blocks differently, more accurate outlines of VOs are obtained. $\psi$-learning, the new classifier that has the advantage over SVM in nonseparable cases, is also employed for both single and multiple VO extraction, and the experimental results demonstrate its superiority.
CHAPTER 8

CONCLUSIONS AND FUTURE WORK

During the past decade, SVM has become a well-known learning machine for its theoretical merit and practical success. Tremendous research efforts have been put forth to make it even more applicable in solving real-world problems. The major work of the dissertation is along the same line. By deriving a set of new formulations, we extend the horizon of SVM in the scenarios when:

1. samples are nonseparable;

2. training samples exhibit partial or ambiguous memberships;

3. it is a multi-category classification problem;

4. rejection of uncharacteristic samples is required;

5. efficient and effective feature selection is needed.

Promising results are obtained, but the work is far from complete to make SVM “perfect”. Each topic on the list above is rich in content, and to address only one of them adequately is a challenging task. For instance, numerous approaches can be found in the literature on the extension of SVM to the multi-category scenario. The problems considered in this dissertation are just a small part of a big picture, which
continues to get extended. In recent years, new methods that are in different research areas but apparently all related to SVM are keeping emerging. Such examples include “support vector tracking”, “support vector clustering”, “kernel kalman filter”. Maybe one day, SVM will have its footprint in every area we can possibly imagine. Until that day, there are still a great deal of work to be done.

Another contribution of this dissertation is the introduction of SVM, or more generally machine learning, to the problem of VO extraction. Aiming at tracking the object of interest at the pixel-wise resolution, VO extraction fundamentally requires the identification for every pixel it belongs to the object or background. Based on this observation, we formulate VO extraction as a classification problem, which by freeing the assumption of motion alleviates the difficulty that the traditional methods encountered in case of motion fluctuations. Moreover, by this formulation, the performance of the extraction is directly linked to the classification accuracy delivered by the classifier. As a result, any advances of machine learning can be directly borrowed to boost the performance.

Real-time is always the ultimate goal if we want the method to be of practical use. So the next step is to further reduce the run time of the algorithm, which fortunately has shown low complexity and good scalability. As a method for video sequence, our method yet relies heavily on spatial information within each frame. The temporal correlation between frames is definitely worth exploiting, by which we believe the real-time goal is within reach.

Besides the VO extraction, there are a great number of applications where the proposed new learning machines can be applied to. One of them is example-based image retrieval, an area which recently has been recognized as a machine learning
problem [100, 101, 102]. In the implementations that adopt relevance feedback, the user is asked to label the images returned by the computer whether they are similar to the image of interest or not, which are to be used to train classifiers such as SVM. The information of the degree of the similarity or dissimilarity, which can usually be obtained by distance measures or indicated by the users themselves, has yet been exploited by the approaches, and $S_{SVM}$ offers this capability. Another important application is bioinformatics. Machine learning, or more specifically SVM, has shown great potential in cancer diagnosis and tumor classification. However, multiple tumor identification imposes new challenges due to the limited medical data and immaturity of the research in multi-class learning. To extend MEMEM to the multi-category scenario and to broaden its application in the area of medical diagnosis would be a very exciting research direction to pursue.
APPENDIX A

PROOF OF EQ. (5.9) AND (5.10)

Recalling that the combination is commutative, we assume SVM\(_i(x) = 1\) for \(1 \leq i \leq l\), and SVM\(_i(x) = -1\) for \(l < i \leq M\), which yields

\[
m = m^+ \oplus m^-,
\]

where \(m^+ = m_1 \oplus \cdots \oplus m_l\), and \(m^- = m_{l+1} \oplus \cdots \oplus m_M\).

It is easy to check that the BPA \(m^+\) only has \(M + 1\) focal elements:

\[
m^+(\pi) = \begin{cases} 
K^+(1 - \beta_1)(1 - \beta_2) \cdots (1 - \beta_{j-1}) & \text{if } \pi = \{H_j\}; \\
\beta_j(1 - \beta_{j+1}) \cdots (1 - \beta_l) & \text{if } \pi = \Omega; \\
0, & \text{otherwise.}
\end{cases}
\]

(A.2)

Here \(K^+\) is the normalization factor to satisfy \(\sum m^+(\pi) = 1\).

The next step is to compute \(m^-\). Let \(\Theta^+ = \{H_1, \ldots, H_l\}, \Theta^- = \{H_{l+1}, \ldots, H_M\}\), and \(2^{\Theta^-}\) denote the power set of \(\Theta^-\). Let

\[
\Theta = \{\Theta^+\} \otimes 2^{\Theta^-} = \{\Theta^+ \cup \theta_i | \theta_i \in 2^{\Theta^-}, 1 \leq i \leq 2^M - 1\}.
\]

(A.3)

It can be shown that only the propositions \(\pi \in \Theta\) have nonzero mass assignments, and therefore \(m^-\) has \(2^{M-l}\) focal elements

\[
m^-(\pi) = \begin{cases} 
K^- \eta_{l+1} \eta_{l+2} \cdots \eta_M, & \text{if } \pi \in \Theta; \\
0, & \text{otherwise,}
\end{cases}
\]

(A.4)
where

\[ \eta_{l+j} = \begin{cases} 
1 - \beta_{l+j} & \text{if } H_{l+j} \in \pi; \\
\beta_{l+j} & \text{if } H_{l+j} \notin \pi.
\end{cases} \tag{A.5} \]

and $K^-$ is the normalization factor.

When combining $m^+$ and $m^-$ by $m = m^+ \oplus m^-$, we encounter two cases. Case \#1 is $l = 0$ which means all the classifiers generate negative responses. Obviously we have $m = m^-$. By using Eq. (5.4), the belief in each hypothesis $\text{Bel}(\{H_i\})$ is obtained as

\[ \text{Bel}(\{H_i\}) = K^-(1 - \beta_i). \tag{A.6} \]

Then the hypothesis $H_{i^*}$ with the highest belief wins

\[ i^* = \arg \max_{i=1,...,M} \text{Bel}(\{H_i\}) = \arg \min_{i=1,...,M} \beta_i = \arg \min_{i=1,...,M} |f_i(x)|. \tag{A.7} \]

Note that all the $f_i(x)$ are negative; therefore, we have

\[ i^* = \arg \max_{i=1,...,M} \text{Bel}(\{H_i\}) = \arg \max_{i=1,...,M} f_i(x), \tag{A.8} \]

which yields Eq. (5.9).

When $l \geq 1$, we have case \#2 where at least one of the SVM classifiers generates positive result, for which the finally combined BPA is

\[ m(\pi) = \begin{cases} 
K^m m^+(\pi), & \text{if } \pi = \{H_i\} \text{ and } i \leq l; \\
K^m m^- (\pi)m^+(\Omega), & \text{if } \pi \in \Theta; \\
0, & \text{otherwise,} \end{cases} \tag{A.9} \]

where $K^m$ satisfies $\sum m(\pi) = 1$.

One can check that in this case $\text{Bel}(\{H_i\}) \geq \text{Bel}(\{H_j\})$ if $i \leq l$ and $j > l$. In other words, the winning class will come from those that generate positive results. Furthermore, for $1 \leq i, j \leq l$ it also satisfies that $\text{Bel}(\{H_i\}) \geq \text{Bel}(\{H_j\})$ iff $\beta_i \geq \beta_j$, or equivalently the winning class $i^*$ should be

\[ i^* = \arg \max_{i=1,...,M} \text{Bel}(\{H_i\}) = \arg \max_{i=1,...,l} \beta_i = \arg \max_{i=1,...,l} f_i(x). \tag{A.10} \]
Since $f_i(x) < 0$ for $l < i \leq M$, Eq. (A.10) can be rewritten as

$$i^* = \arg \max_{f_i(x) \geq 0} f_i(x) = \arg \max_{i=1,\ldots,M} f_i(x),$$

(A.11)

which yields Eq. (5.10).
BIBLIOGRAPHY


