Kernel Optimization for Support Vector Machines: Application to Speaker Verification

by

Andrew Oliver Hatch

B.E.E. (University of Dayton) 1998
M.S. (University of California, Berkeley) 2001

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Engineering-Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Nelson Morgan, Chair
Professor Michael Jordan
Professor Alper Atamturk

Fall 2006

The dissertation of Andrew Oliver Hatch is approved.

Chair	Date
	Date
	Date

University of California, Berkeley

Fall 2006

Kernel Optimization for Support Vector Machines: Application to Speaker Verification

Copyright © 2006

by

Andrew Oliver Hatch

Abstract

Kernel Optimization for Support Vector Machines: Application to Speaker Verification

by

Andrew Oliver Hatch

Doctor of Philosophy in Engineering-Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Nelson Morgan, Chair

In this dissertation, we examine the problem of kernel optimization for binary classification tasks where the training data are partitioned into multiple, disjoint classes. The dissertation focuses specifically on the field of speaker verification, which can be framed as a one-versus-all (OVA) decision task involving a target speaker and a set of impostor speakers.

The main result of this dissertation is a new framework for optimizing *generalized linear kernels* of the form, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{R} \mathbf{x}_2$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are input feature vectors, and $\mathbf{R}$ is a positive semidefinite parameter matrix. Our framework is based on using first and second-order statistics from each class (i.e., speaker) in the data to construct an upper bound on classification error in a linear classifier. Minimizing this bound leads directly to a new, modified formulation of the *1-norm, soft-margin support vector machine* (SVM). This modified formulation is identical to the conventional SVM developed by Vapnik, except that it implicitly prescribes a solution for the $\mathbf{R}$ parameter matrix in a generalized linear kernel. We refer to this new, modified SVM formulation as the *adaptive, multicluster SVM* (AMC-SVM). Unlike most other kernel learning techniques in the literature, the AMC-

SVM uses information about clusters that reside within the given target and impostor data to obtain tighter bounds on classification error than those obtained in conventional SVM-based approaches. This use of cluster information makes the AMC-SVM particularly well-suited to tasks that involve binary classification of multiclass data—for example, the speaker verification task—where each class (i.e., speaker) can be treated as a separate cluster.

In OVA training settings, we show that the AMC-SVM can, under certain conditions, be formulated to yield a single, fixed kernel function that applies universally to any choice of target speaker. Since this kernel function is linear, we can implement it by applying a single linear feature transformation to the input feature space. This feature transformation performs what we refer to as *within-class covariance normalization* (WCCN) on the input feature vectors. The dissertation describes a set of experiments where WCCN yields large reductions in classification error over other normalization techniques on a state-of-the-art SVM-based speaker verification system.

Professor Nelson Morgan
Dissertation Committee Chair

This dissertation is dedicated to my wife, Celeste, to my loyal friends, and to Gautam and Sigi's cat, Leo, who sat on my lap through much of its writing.

Contents

List of Figures

List of Tables

Acknowledgements

This dissertation would not have been possible without the generous support of various individuals and organizations. I would first of all like to thank my faculty advisor, Nelson Morgan, for his guidance and support throughout my graduate school career. Nelson Morgan is the director of the International Computer Science Institute (ICSI) in Berkeley, California. ICSI is affiliated with the University of California, Berkeley, which is where I have conducted my Ph.D. research. I would also like to thank my group leaders, Barbara Peskin and Naghmeh (Nikki) Mirghafori, for many interesting and helpful discussions on research within the field of speaker verification. Since the fall of 2003, Barbara and Nikki have both served as group leaders for ICSI's speaker verification team. Both were also instrumental in securing funding for my research. The funding itself was provided by the National Science Foundation under grant No. 0329258. Special thanks are also due to George Doddington, who advised me on a number of research trends within speaker verification. I am especially grateful to my primary collaborator, Andreas Stolcke of SRI International, for his intellectual and moral support, and for his constant interest and enthusiasm for my work. I would also like to thank Luciana Ferrer and Sachin Kajarekar of SRI, who both provided critical support and infrastucture for ICSI's speaker verification research. Over the past few years, Sachin has collaborated with me on various projects within the field of speaker verification. I would also like to thank Laurent El Ghaoui for his help in proving Theorem 14, which states that the *adaptive, multicluster SVM* can be posed as a second-order cone program (see Chapter 7). Special thanks are due to the members of my dissertation committee: Michael Jordan, Alper Atamturk, and my advisor, Nelson Morgan, for their support and assistance in completing this work. Finally, I would like to thank the other current and former members of the ICSI speaker verification group with whom I have had the pleasure of working: Lara Stoll, Howard Lei, Mary Knox, Kofi Boakye, and Dan Gillick.

Curriculum Vitæ

Andrew Oliver Hatch

Education

1998	University of Dayton
	B.S., Electrical Engineering
2001	University of California, Berkeley
	M.S., Electrical Engineering and Computer Science
2006	University of California, Berkeley
	Ph.D., Electrical Engineering and Computer Science

Personal

Born	February 13, 1975, Dayton, Ohio

Chapter 1

Introduction

One of the central problems in the study of support vector machines (SVMs) is kernel selection—that is, the problem of choosing or optimizing a kernel function for a particular task and dataset. In the following dissertation, we consider the problem of kernel optimization for binary classification tasks where the training data are partitioned into multiple, disjoint classes. We focus specifically on the field of speaker verification, which can be framed as a one-versus-all (OVA) decision task involving a target class (i.e., speaker), and an impostor class composed of a pooled set of impostor speakers. The dissertation shows how information about individual classes (i.e., speakers) in the data can be used to construct upper bounds on classification error. By minimizing these upper bounds, we obtain a new framework for training affine classifiers. This framework also leads to various techniques for training kernel functions for support vector machines. We describe a set of experiments in which our approach yields substantial improvements in classification error when compared with other, existing techniques for training kernel functions.

The main result of this dissertation is a general framework for performing binary classification in multiclass settings with affine decision functions. This framework leads directly to a modified formulation of both the hard-margin and the 1-norm soft-margin support vector machines (SVMs). These modified SVMs are identical to the conventional SVM

formulation in *Vapnik* [1995], except that they implicitly learn kernel functions of the form, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{R} \mathbf{x}_2$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are input feature vectors, and $\mathbf{R}$ is a positive semidefinite matrix. We refer to kernel functions of this form as *generalized linear kernels*.

Under various conditions, our framework leads to generalized linear kernels of the form $\mathbf{R} = \mathbf{C}_W^{-1}$, where $\mathbf{C}_W$ is the expected *within-class covariance matrix* over all classes (i.e., speakers) in the training data. This parameterization applies universally to any choice of target class (i.e., speaker). Because the kernel function is linear, we can implement it by applying a single linear feature transformation to the input feature space. This feature transformation performs what we refer to as *within-class covariance normalization* (WCCN) on the input feature vectors. In Chapters 6 and 8, we describe a set of experiments where WCCN yields large reductions in classification error over other normalization techniques. The WCCN approach forms one of the main components of our dissertation.

In Chapter 7, we derive a more general formulation of WCCN where the kernel function is adapted to the given choice of target set and impostor set. We refer to this formulation as the *adaptive, multicluster SVM* (AMC-SVM), since it assigns a separate weight parameter to every class (i.e., cluster) in the training data. Unlike other kernel learning techniques in the literature (e.g., multiple kernel learning (*Lanckriet et al.* [2004]; *Bach et al.* [2004]) and hyperkernels (*Ong et al.* [2003]), the AMC-SVM uses information about clusters that reside *within* the given target and impostor data to obtain tighter bounds on classification error than those obtained in conventional SVM-based approaches. This use of cluster information makes the adaptive, multicluster SVM particularly well-suited to tasks that involve binary classification in multiclass settings—for example, the speaker verification task, where each class (i.e., speaker) can be treated as a separate cluster.

The dissertation is organized as follows: We begin, in Chapter 2, by describing the speaker verification task and by summarizing the current state-of-the-art in the field, particularly as it applies to support vector machines. This is followed by a brief overview of support vector machines (SVMs) in Chapter 3. Chapter 4 provides a brief summary of

related work within the field of kernel optimization. Chapters 5 through 7 describe the theoretical framework behind WCCN and the adaptive multicluster SVM. These chapters also describe a set of experiments where we compare these techniques with other state-of-the-art techniques for training kernel functions. Finally, we describe a practical procedure for applying WCCN to high-dimensional datasets in Chapter 8. In this chapter, we also describe the latest results obtained from using WCCN on an SVM-based speaker verification task. The results show that WCCN yields significant improvements over other state-of-the-art techniques for performing kernel optimization on speaker verification tasks.

Chapter 2

Speaker Verification

The following chapter gives a brief introduction to the field of speaker verification. This includes a description of the various models and feature sets that are typically used to train speaker verification systems. In the following section, we begin with a general overview of the speaker verification problem.

2.1 Problem Definition

Unlike speaker identification, where the goal is to associate a given speech utterance with a specific speaker, the goal in most NIST-defined speaker verification tasks—which are the tasks that we will consider for this research—is to determine whether or not a given speech utterance belongs to a given speaker. Thus, speaker verification is a binary classification problem. We will focus on a set of NIST-defined tasks for performing *text-independent* speaker verification, where the input speech is not constrained to any specific set of words or phrases. In these tasks, the speaker verification system is presented with a set of speech utterances or *conversation sides*. These conversation sides constitute one side of a two-way telephone conversation and are typically about 2.5 minutes in length. The speaker verification system is presented with a limited number of conversation sides

(usually between 1 and 8) from the given *target speaker*, along with a test conversation side, which we refer to as the *test segment* or *test utterance*. The system is then asked to determine whether or not the test utterance belongs to the target speaker. Typically, the speaker verification system is allowed to use any number of conversation sides taken from so-called *impostor speakers*—that is, speakers who don't appear in the test data or as target speakers—to facilitate in making this decision. This pooled set of impostor conversation sides is referred to as the *background data* or as the *impostor data*.

2.1.1 Prototypical System

Figure 2.1 shows a high-level diagram of a prototypical speaker verification system. The system uses a set of target utterances and a set of impostor or *background* utterances to train a one-versus-all (OVA) speaker model for speaker i. In the figure, we use the notation, $\mathbf{X}_i$ to represent the set of training utterances that belong to speaker i. These utterances constitute the "positive examples" for the given speaker. Similarly, we refer to all utterances in $\mathbf{X}_j$ where $j \neq i$ as the negative or *impostor* examples for speaker i. The system in Figure 2.1 uses the target examples and the impostor examples for speaker i to train a speaker model, which we represent as the function f_i. The speaker model performs a mapping from an L-dimensional *input space* into the space of real numbers:

$$f_i \ : \ \mathbb{R}^L \rightarrow \mathbb{R}.$$

We refer to the scalar value $f_i(\mathbf{x})$ as the output *score* obtained for utterance $\mathbf{x}$ on speaker model i. This score is used to form the output hypothesis $\hat{y}_i(\mathbf{x})$, which indicates whether or not the system believes that $\mathbf{x}$ belongs to target speaker i. This hypothesis is determined as follows:

$$\hat{y}_i(\mathbf{x}) = \begin{cases} 1, & \text{if } f_i(\mathbf{x}) \geq 0, \\ -1, & \text{if } f_i(\mathbf{x}) < 0. \end{cases} \tag{2.1}$$

The term $\hat{y}_i(\mathbf{x})$ represents the hypothesized binary label for $\mathbf{x}$ given target speaker i. Here, a value of 1 represents a "positive" decision, where $\mathbf{x}$ is deemed to belong to speaker i.

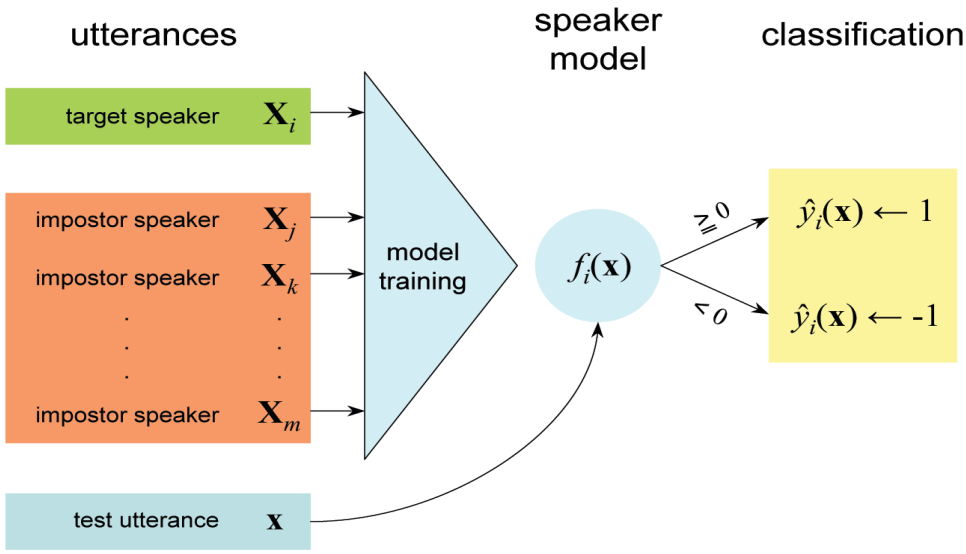

Figure 2.1. High-level diagram of a typical speaker verification system. Here, $\mathbf{X}_i$ represents the set of training utterances that belong to target speaker i. The functions f_i and y_i represent the corresponding speaker model and decision function.

Conversely, a value of -1 represents a "negative" decision, where $\mathbf{x}$ is said to belong to the impostor set. We use $y_i(\mathbf{x})$ to represent the *true* label of $\mathbf{x}$ given target speaker i:

$$y_i(\mathbf{x}) = \begin{cases} 1, & \text{if } x \text{ belongs to speaker } i, \\ -1, & \text{otherwise.} \end{cases}$$

If $\hat{y}_i(\mathbf{x})$ is equal to $y_i(\mathbf{x})$, then we say that $\mathbf{x}$ has been correctly classified for the given target speaker i. Otherwise, we say that $\mathbf{x}$ has been misclassified.

Signals and Feature Sets

Given the many sources of speaker-specific information within speech (e.g., prosody, pitch, word usage, speaking rate, etc.), most state-of-the-art speaker verification systems rely on multiple signals and feature representations to encode each conversation side or *utterance*. For example, a speaker verification system might use a set of *acoustic* features—that is, features based on short-time Fourier representations of the input acoustics (e.g., Mel-frequency cepstral coefficients (MFCCs) (*Kajarekar et al.* [2005b])—along with feature

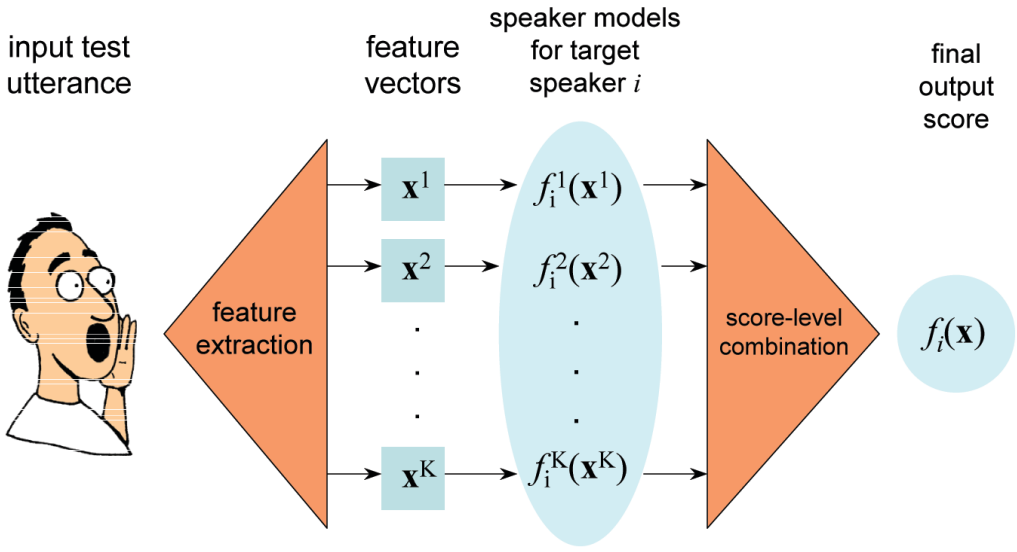

Figure 2.2. High-level diagram of a speaker verification system that performs score-level system combination.

sets based on relative n-gram frequencies of words (*Doddington* [2001]) and of phonemes (i.e., sub-word linguistic units) (*Andrews et al.* [2002]; *Campbell et al.* [2003]; *Hatch et al.* [2005]), and a fourth set of features based on prosodic events (*Kajarekar et al.* [2003]), to represent each conversation side. We will provide a more detailed description of these signals and feature sets in Section 2.2.

In general, the goal in speaker verification is to capture as much speaker-specific information as possible within each conversation side by using a diverse range of signals and feature sets. In many contemporary speaker verification systems, these feature sets are used to train a set of individual speaker models, where one speaker model is trained for every feature set. Each speaker model generates its own output scores, and these scores are later combined to generate a final output score for a given utterance $\mathbf{x}$. A system such as this is shown in Figure 2.2. In the above figure, we use the notation, $\mathbf{x}^j$, to denote the feature vector for utterance $\mathbf{x}$ drawn from signal (i.e., feature set) j. The system in Figure 2.2 trains a set of functions of the form, $f_i^j : \mathbb{R}^{L_j} \rightarrow \mathbb{R}$, where f_i^j represents the *speaker model* corresponding to target speaker i and feature set j. This speaker model maps feature vectors

from some L_j-dimensional feature space (i.e., the feature space that corresponds to the jth feature set) into a real-valued output *score*. The output scores for a particular *test-target pair*—for example, the pair $(\mathbf{x}, i)$, where $\mathbf{x}$ represents a test utterance and i represents a target speaker model—are then "combined" in some way to arrive at a final output score, $f_i(\mathbf{x})$. We will briefly discuss techniques for combining scores in Section 2.7. Finally, we use the decision rule in (2.1) to obtain the output hypothesis, $\hat{y}_i(\mathbf{x})$.

The system shown in Figure 2.2 depicts what we refer to as *score-level combination*, where the input signals and feature sets are combined at the level of output scores. Systems such as this are quite common in the field of speaker verification and often perform quite well on system evaluations. For example, the two top-performing systems in the 2005 NIST Speaker Recognition Evaluation both used a score-level combination strategy (*Mirghafori et al.* [2005]; *Ferrer et al.* [2005a]). Score-level combination provides a convenient means of "combining" feature sets in a single system, because it treats all feature sets as though they were independent of one-another, conditional on their output scores. The cost of this conditional indepedence assumption is that the system in Figure 2.2 is incapable of modeling interdependencies between signals that appear only at the feature level. Unless the signals are truly independent conditional on their output scores, score-level combination can lead to reductions in classification accuracy below what might otherwise be achievable.

In this dissertation, we investigate techniques for performing what we refer to as *feature-level combination*, where various signals and feature sets are "combined" into a single feature representation prior to training speaker models. A system that performs feature-level combination is shown in Figure 2.3. In this system, the input feature sets are first fed into a feature normalization module, which produces a single, combined feature set at its output. This output feature set is then used to train a single speaker model. Here, the idea is to design a feature normalization module that transforms and combines the input feature sets in some optimal (or near optimal) way for the purpose of training speaker models. The question of how to transform or combine feature sets is central to this dissertation and will be discussed in much greater detail in subsequent chapters.

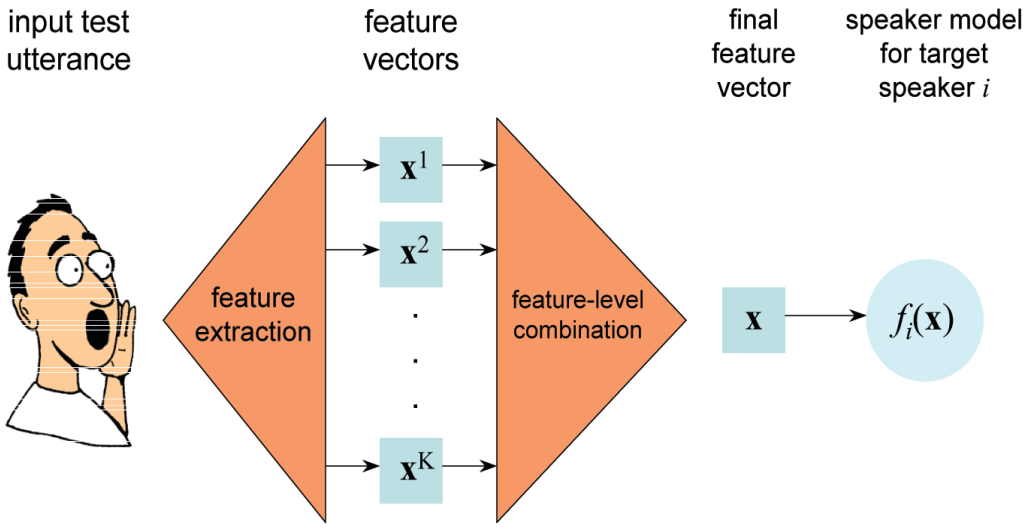

Figure 2.3. High-level diagram of a speaker verification system that performs feature-level system combination.

2.2 Classification Paradigms for Speaker Verification

The previous section describes the speaker verification problem and provides a general overview of a typical speaker verification system. In Section 2.3 and in Section 2.4, we discuss the two main classification paradigms used in contemporary speaker verification. These include Maximum Likelihood (ML) classification, which involves modeling the conditional probability density function (pdf) of the utterance space given either the target speaker or the impostor speakers. Our discussion of ML classification will focus mainly on Gaussian mixture models (GMMs), which are used to estimate pdfs. GMMs have traditionally been among the most widely-used tools for modeling speaker characteristics, and one of the most successful for performing speaker verification. We will also provide a brief overview of other ML classification techniques that have played a significant role in speaker verification—for example, hidden Markov models (HMMs) and n-gram models.

Section 2.4 is devoted to the more recent paradigm of SVM-based speaker verification, where SVMs are used to train speaker models. In recent years, support vector machines have become one of the most important and widely-used classification techniques within the

field of speaker verification. Section 2.4 provides a high-level overview of the various feature sets (a.k.a. "systems") that have been developed for SVM-based speaker verification. A detailed description of support vector machines is provided in Chapter 3.

2.3 Maximum Likelihood Classification for Speaker Verification

Until recently, the field of speaker verification has been largely dominated by maximum likelihood (ML) techniques for performing classification. The primary feature of these techniques is that they attempt to model the conditional *probability density function* (pdf) of the utterance space for both the target speaker and the impostor speakers. In the ML framework, a given test example $\mathbf{x}$ is classified as belonging to the class with the highest conditional pdf for $\mathbf{x}$. We represent the conditional pdf of $\mathbf{x}$ as $f_{\mathbf{x}|y_i}$, where $y_i \in \{-1, 1\}$ denotes the particular *model*. By convention, a value of 1 for y_i denotes the model for target speaker i, and a value of -1 denotes the model for the impostor set. Given $f_{\mathbf{x}|y_i}$ for $y_i \in \{-1, 1\}$, we can form the log-likelihood ratio for $\mathbf{x}$ given target speaker i as follows:

$$LLR(\mathbf{x}\ ;\ i) \triangleq \log\left(\frac{f_{\mathbf{x}|y_i=1}(\mathbf{x})}{f_{\mathbf{x}|y_i=-1}(\mathbf{x})}\right).$$

In ML classification, the log-likelihood ratio for test utterance $\mathbf{x}$ given target speaker i is typically used to define the corresponding output *score*, $f_i(\mathbf{x})$:

$$f_i(\mathbf{x}) \triangleq LLR(\mathbf{x}\ ;\ i).$$

Given the above assignment for $f_i(\mathbf{x})$, we note that if $f_i(\mathbf{x})$ is greater than zero, then ML classification yields the output hypothesis, $\hat{y}_i(\mathbf{x}) = 1$. Similarly, if $f_i(\mathbf{x})$ is less than zero, then we have the hypothesis, $\hat{y}_i(\mathbf{x}) = -1$. Thus, we can use the decision rule in (2.1) to obtain the output hypothesis, $\hat{y}_i(\mathbf{x})$.

2.3.1 GMM-based Classification

The most common form of ML estimation in speaker verification involves using Gaussian mixture models (GMMs) to estimate the pdfs of the utterance space. The GMM-based approach for speaker verification is largely inherited from the field of automatic speech recognition, where GMMs are used to model pdfs for phonemes and for other speech units. In most GMM-based systems, the input conversation sides are divided into short-time "frames" of speech. These frames are typically about 30ms in length and are sampled at 10ms intervals. A set of mel-frequency cepstral coefficients (MFCCs) are extracted for each speech frame (*Davis and Mermelstein* [1980]). For example, the state-of-the-art GMM system documented in *Kajarekar et al.* [2005b] extracts a total of 19 MFCCs for each 30ms frame of input speech. The MFCCs are concatenated with a set of so-called *delta* features, which represent the first and second differences between the MFCCs in the current frame and those in the adjacent frames. Further details on feature extraction for GMM-based speaker verification can be found in *Reynolds et al.* [2000]; *Kajarekar et al.* [2005b].

After performing feature extraction, the system in *Kajarekar et al.* [2005b] uses a GMM to model the unconditional pdf of $\mathbf{x}$, which we represents as $f_{\mathbf{x}}$. The unconditional pdf of $\mathbf{x}$ is often referred to as the *speaker-independent acoustic model*. Given the large number of speakers in most speaker verification training sets, $f_{\mathbf{x}}$ is typically used as a universal estimate for the conditional pdf of the impostor set, $f_{\mathbf{x}|y_i=-1}$. In other words, we use the approximation, $f_{\mathbf{x}|y_i=-1} \approx f_{\mathbf{x}}$ for all i. The conditional pdf of target speaker i is typically estimated by adapting the speaker-independent acoustic model (i.e., $f_{\mathbf{x}}$) to the training data for that speaker. We refer to this pdf as a *speaker-dependent acoustic model* for speaker i. Common techniques for performing adaptation include *maximum a posterior* (MAP) adaptation (*Gauvain and Lee* [1994]) and *maximum-likelihood linear regression* (MLLR) (*Leggetter and Woodland* [1995]). After obtaining $f_{\mathbf{x}|y_i}$ for $y \in \{-1, 1\}$, we can use the equations prescribed in the previous section to compute log-likelihood ratios and output scores for target speaker i.

2.3.2 HMM-based Classification

GMM-based systems are typically among the top-performing systems on speaker verification tasks. However, GMMs are often criticized for their inability to capture sequence information in speech. For example, since words usually span many frames, GMMs tend to be poorly suited for modeling differences in word usage (idiolect) between speakers. Indeed, the GMM-based approach is often referred to as a "bag of frames," because GMMs are invariant to the sequence in which frames appear in a conversation side. One way to address this issue is to use hidden Markov models (HMMs) to model the pdfs and conditional pdfs of the utterance space. Unlike GMMs, HMMs incorporate sequence information into their likelihood estimates. HMMs are also, by far, the most widely-used tool for modeling speech within the field of automatic speech recognition (*Rabiner and Juang* [1993]).

HMMs typically work well on text-*dependent* speaker verification tasks, where the speakers are instructed to read a given word or phrase (*Rosenberg et al.* [1990, 1991]). HMMs have also achieved some success on text-independent tasks like the tasks that we consider in this dissertation. For example, Boakye and Peskin used word-constrained HMMs to build a text-independent speaker verification system in *Boakye and Peskin* [2004]. This work played a significant role in ICSI's submission for the NIST 2005 speaker recognition evaluation, in which ICSI achieved the second-best results out of all participating sites on the so-called "common condition" (*Mirghafori et al.* [2005]).

2.3.3 N-gram Models

Another effort at moving beyond the standard GMM-based paradigm is to explicitly model sequences of phones and/or words used by speakers. This line of research was pioneered by Doddington, who used counts of word n-grams to model speaker-specific patterns of word usage (*Doddington* [2001]). In Doddington's paper, the counts are obtained from the output of an automatic speech recognition (ASR) system. As in the GMM-based approach, Doddington's approach involves computing the log-likelihood ratio (LLR) for every pair of

test utterance and target speaker. Doddington used the following equation to compute the LLR for test conversation side $\mathbf{x}$ and target speaker model k:

$$LLR(\mathbf{x}\,;\,k) = \sum_{i=1}^{M} p(d_i|\mathbf{x}) \log \frac{p(d_i|spk_k)}{p(d_i|bkg)} \tag{2.2}$$

Here, $p(d_i|\mathbf{x})$, $p(d_i|spk_k)$, and $p(d_i|bkg)$ refer to the prior probability (or equivalently, the *relative frequency*) of word n-gram d_i within conversation side $\mathbf{x}$, speaker model k, and within the background model, respectively. In principle, this equation can be applied not only to word n-grams, but to any n-gram unit—for example, n-grams based on speech phonemes. As in the GMM-based approach, the LLRs defined by the above equation are used as output scores in a speaker verification system.

An approach similar to Doddington's has also been used to model patterns of phoneme usage in speech. This line of research, which is sometimes referred to as *phonetic speaker recognition*, was introduced by Andrews et al., who used relative frequencies of phone n-grams derived from a speech recognizer to capture patterns in an individual's speech (*Andrews et al.* [2001, 2002]). This work was subsequently extended in various papers, such as the work of the "SuperSID" team at the JHU 2002 Summer Workshop (*Jin et al.* [2003]; *Navratil et al.* [2003]; *Klusacek et al.* [2003]; *Reynolds et al.* [2003]). In 2003, Campbell et al. used support vector machines (SVMs) to train phonetic speaker models (*Campbell et al.* [2003]). Subsequent improvements to this paradigm are described by *Hatch et al.* [2005]. The n-gram models described here are typically less effective than GMM-based systems on speaker verification tasks. However, these models tend to yield significant improvements in classification error when combined with GMM systems (*Hatch et al.* [2005]).

2.4 Speaker Verification with Support Vector Machines

Until fairly recently, the GMM-based approach described in Section 2.3 was widely regarded as the standard in state-of-the-art speaker recognition technology. However, over the past few years, the field of speaker recognition has essentially been transformed by sup-

port vector machines (SVMs), which now play an important role in most high-performance speaker recognition systems. Support vector machines are specifically designed for binary classification tasks, and they tend to work particularly well on tasks where the dimensionality of the feature space is large compared with the total number of training examples. These properties make SVMs well-suited for speaker verification tasks, where the input feature spaces are large (typically between 10000 to 100000 features), and the number of training examples is comparatively small (typically between one and eight target examples and several thousand impostor examples).

Most SVM-based speaker recognition systems train a separate SVM for every combination of target speaker and feature set. Typically, each SVM is trained in a one-versus-all (OVA) setting, where the conversation sides of the target speaker are used as the positive examples and the conversation sides of the impostor speakers are used as the negative examples. The SVM-based *scoring function* for a given target speaker i is used to define the corresponding speaker model, f_i. In the following sections, we provide an overview of SVM-based speaker recognition and describe some of the most widely-used feature sets within this field. A more in-depth introduction to SVMs is provided in Chapter 3. We also provide an overview in Chapter 4 of the various feature transformations and kernel functions used in SVM-based speaker verification.

2.5 Feature Sets for SVM-based Speaker Verification

The following section provides a brief description of some of the most commonly-used feature sets in SVM-based speaker verification.

2.5.1 Cepstral Features

One of the most widely-used feature sets in SVM-based speaker verification is the so-called *cepstral SVM* system, which was first introduced by *Campbell* [2001]. (Note that we

use the terms, "system" and "feature set," interchangeably in this section). Cepstral SVM systems are essentially the SVM-based counterparts of the Gaussian mixture model (GMM) systems described in Section 2.3. A typical cepstral SVM system extracts ∼13 cepstral features per speech frame. An intermediate set of features is then formed by computing various derivatives of the cepstral features, which are transformed into a polynomial vector, typically of degree 3 or less. The polynomial vectors are then averaged over an entire conversation side to arrive at the final feature representation (*Campbell* [2001]). Most state-of-the-art implementations of the cepstral SVM have in excess of 10000 features per feature vector. In a comparison of SVM-based speaker verification systems (i.e., feature sets), the lowest error rates are typically achieved by cepstral SVM systems and by MLLR-SVM systems, which we describe next.

2.5.2 MLLR Features

MLLR-SVM systems use feature vectors composed of transform coefficients obtained from a *maximum-likelihood linear regression* (MLLR) (*Leggetter and Woodland* [1995]). The MLLR approach involves training a linear transformation to map the means of a speaker-independent GMM into a new set of means for a given conversation side. This approach was originally developed to transform speaker-independent acoustic models (i.e., GMMs) into speaker-dependent models for speech recognition tasks. However, Stolcke et al. recently showed that MLLR can also be used to capture speaker-specific information for performing speaker verification. The MLLR-SVM systems described in *Stolcke et al.* [2005, 2006] use MLLR transform coefficients to construct feature vectors for performing speaker verification with SVMs. A typical MLLR-SVM system may have in excess of 12000 features per feature vector. The MLLR-SVM systems that have been reported so far in the literature have been shown to yield superior results over most other feature sets. We note that MLLR-SVM systems form the basis for many of the experiments that we report in Chapters 6 and 8.

2.5.3 N-gram Features

Another common paradigm in SVM-based speaker verification is *n-gram* or *count-based* feature extraction, where each feature represents the relative frequency of some n-gram event (e.g., a word or a phoneme). Feature sets based on relative frequencies of n-grams were previously described in Section 2.3.3 in the context of ML classification. In recent years, these feature sets have also played a significant role in SVM-based speaker verification. For example, a number of papers have been written on the use of relative frequencies of phone (i.e., phoneme) n-grams as features for SVM-based speaker verification (*Campbell et al.* [2003]; *Hatch et al.* [2005]). Phone n-gram systems typically use anywhere from 2000 to 50000 n-gram based features per feature vector. SVM-based phone n-gram systems consistently outperform their ML-based counterparts. However, phone n-grams have, so far, failed to achieve the same level of classification accuracy as other SVM-based systems—most notably, the cepstral and MLLR-SVM systems described in the previous subsections. Phone n-gram systems have also failed to yield significant improvements when combined at the score-level with state-of-the-art speaker verification systems.

Other types of n-gram systems include the word n-gram system described in *Kajarekar et al.* [2005a]. This system forms the SVM-based analog of the word n-gram system developed by *Doddington* [2001]. As with phone n-grams, word n-grams tend to give better results in SVM-based systems than in ML-based systems. The word n-gram system described in *Kajarekar et al.* [2005a] performs poorly when tested as a stand-alone system, but yields significant improvements when combined at the score-level with a state-of-the-art speaker verification system.

Other notable feature representations include so-called *SNERF n-grams*, which model n-gram frequencies of prosodic events at the syllable level (*Shriberg et al.* [2004]; *Kajarekar et al.* [2003]). These feature sets typically have over 30000 features per feature vector, where each feature represents the frequency of a particular n-gram event. Although less effective in isolation than most "acoustic" feature sets (e.g., cepstral SVM and MLLR-SVM systems),

SNERF n-grams tend to yield significant improvements in overall accuracy when combined at the score-level with other systems.

2.6 Score Normalization

We can use the techniques described in the preceding sections to train speaker models for all target speakers in a given training set. In general, these speaker models will tend to produce scores that discriminate between utterances that belong to the given target speaker and utterances that do not. However, because the speaker models are trained independently of one-another, the resulting output scores may be biased. More importantly, the output scores may have biases that differ significantly conditional on the speaker model for which they were computed. For example, the scores for the positive and negative trials of speaker model i (i.e., the trials where test utterances belong to speaker i and the trials where they do not) may be centered around 0.5 and 0, respectively, while the scores for the positive and negative trials of speaker model j are centered around 0 and -0.5. Similarly, an output score may be biased conditional on the test utterance for which it was computed. In either case, the resulting output scores will not be "aligned" properly—especially if we plan to use a single, fixed score threshold over all speaker models and test utterances. To correct for this, most speaker verification systems apply various normalizations to the output scores.

The two most common forms of score normalization for speaker verification are zero-normalization (ZNORM) (*Reynolds* [1997]) and test-normalization (TNORM) (*Auckenthaler et al.* [2000]). The ZNORM approach involves computing *impostor scores*—that is, scores for *impostor* or *negative* trials where the given test utterance does not belong to speaker i—on some set of background data for every speaker model. The output scores are then shifted and scaled in such a way that the impostor scores have a fixed sample mean and sample variance for every speaker model. The TNORM approach is similar to ZNORM except that it normalizes the output scores across test utterances instead of across models. TNORM can be applied to a given test utterance $\mathbf{x}$ by first scoring $\mathbf{x}$ with a set of "impostor

models" (i.e., speaker models that do not "belong" to $\mathbf{x}$). This gives us a set of impostor scores for $\mathbf{x}$. We then shift and scale all output scores obtained with $\mathbf{x}$ in such a way that the impostor scores for $\mathbf{x}$ have a fixed sample mean and sample variance. The TNORM and ZNORM approaches tend to yield significant improvements in classification performance on most speaker verification tasks. Note that these techniques can also be combined—that is we can apply ZNORM after TNORM or vice-versa—to perform score normalization.

2.7 System Combination

In Section 2.1.1, we briefly described the problem of combining multiple feature sets and information sources in a single speaker verification system. This problem, which we refer to as *system combination*, is typically handled by combining various individual systems and feature sets at the score-level. A diagram of this type of system combination is provided in Figure 2.2. In many state-of-the-art speaker verification systems, the final output scores are computed as a weighted sum of the scores of the individual systems:

$$f_i(\mathbf{x}) = \sum_j \sigma_j f_i^j(\mathbf{x}).$$

Here, σ_j represents the weight of the jth *subsystem*—that is, the jth feature set. Various techniques can be used to train these weights. For example, many top-performing systems use a single-layer perceptron to train weights for the various subsystems (*Kajarekar et al.* [2005b]; *Mirghafori et al.* [2005]). Support vector machines (SVMs) have also recently been used for this purpose (*Garcia-Romero et al.* [2003]). Other notable techniques for performing score-level combination for speaker verification are described in *Ferrer et al.* [2005b].

The techniques decsribed above provide a convenient and relatively straightforward means of combining feature sets (i.e., subsystems) into a single system. However, as described in Section 2.1.1, one potential problem with score-level combination is that the feature sets are assumed to be independent of one-another conditional on the output scores. In other words, the feature sets are only allowed to "interact" with each other at the score

level, which means that most task-relevant interdependencies between the feature sets—if they exist in the first place—will be lost. Viewed from the perspective of information theory, score-level combination can have the effect of reducing the channel capacity of the classification system, which essentially places a lower bound on classification error.

One of our goals in this thesis proposal is to address this problem by performing what we refer to as *early combination* or *feature-level combination*—that is, training speaker models on one single combined set of features rather than combining feature sets at the level of output scores. The concept of feature-level combination is illustrated in Figure 2.3. In the context of SVM-based classification, early combination boils down to the question of how to select or learn a single *kernel* that can handle features from multiple knowledge sources. This topic will be addressed in greater detail in subsequent chapters. First, we provide a brief introduction to support vector machines in Chapter 3.

Chapter 3

Support Vector Machines

The following chapter provides an overview of the learning system known as the *support vector machine* (SVM). We begin the chapter with some background on SVMs, including a brief introduction to the related concepts of *structural risk minimization*, *VC dimension*, and *optimal margin classifiers* (*Boser et al.* [1992]). This is followed in Sections 3.2 and 3.3 with descriptions of the so-called *hard-margin* and *soft-margin* SVMs. Throughout this chapter, we refer to various concepts from the field of convex optimization. These concepts include, for example, *linear programs* (LPs), *quadratic programs* (QPs), *convex duality*, and the notion of a convex optimization problem. A thorough description of these concepts can be found in the book by *Boyd and Vandenberghe* [2004].

3.1 Background

The term *support vector machine* refers to a system for learning functions of the form, $f(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) + b$, where $\mathbf{x}$ represents an input vector, $\mathbf{w} \in \mathbb{R}^N$ and $b \in \mathbb{R}$ represent trained parameters, and $\Phi : \mathcal{X} \to \mathcal{F}$ represents a mapping from the input space $\mathcal{X}$ to some feature space $\mathcal{F}$. Typical applications of SVMs include regression, where $f(\mathbf{x})$ is trained to map $\mathbf{x}$ to some set of desired output values, and binary classification, where the set $\{\mathbf{x} : f(\mathbf{x}) = 0\}$ is

used to define a separating hyperplane between two classes in $\mathcal{F}$. Support vector machines are based on various concepts of statistical learning theory that have been developed over the course of several decades by Vladimir Vapnik and his associates. These include the concept of *structural risk minimization*, which involves constructing and minimizing upper bounds on the probability of misclassifying future data. We refer to the probability of misclassification as the *risk* for a given dataset and classifier. Structural risk minimization differs significantly from the more conventional approach of *empirical risk minimization*, where classifiers are trained to minimize the empirical error incurred on a training set. (In practice, empirical risk minimization is often performed using gradient descent along with a *stopping criterion*, where training is halted once the classification error stops decreasing on a cross-vaidation dataset). Vapnik and his associates introduced various upper bounds on risk that depend on both the empirical risk—that is, the error incurred on a given training set—and on various notions of the *capacity* of a learning system. Within the field of classification, the term *capacity* essentially refers to complexity of a given family of decision boundaries. In general, greater capacity corresponds with greater modeling power. Thus, increased capacity can lead to reductions in the empirical risk. However, greater capacity can also increase the risk of overfitting. A sensible strategy for training classifiers therefore involves minimizing the empirical risk while also limiting the capacity (or vice-versa). This strategy forms the basic intuition behind *structural risk minimization* and behind the support vector machine.

3.1.1 VC Dimension

Vapnik's upper bounds on risk lead to a particular notion of model capacity called the Vapnik-Chervonenkis or "VC" dimension. The VC dimension of a family F of classifiers is defined as the maximum number N of non-colinear examples which, for any set of labels in $\{-1, 1\}^N$, can be perfectly separated by a function in F. For example, let us define F to be the set of all possible hyperplanes in $\mathbb{R}^2$. Any set of three non-colinear examples can be separated by at least one function in F. However, the same is not true for any set of

four examples. Thus, the VC dimension of F is three, in this case. We can extrapolate this example to show that if F is composed of all hyperplanes in $\mathbb{R}^n$, then the VC dimension of F is $n+1$. In general, we would like to minimize the VC dimension of F while at the same time minimizing empirical risk. Vapnik and his colleagues use this strategy to motivate the concept of *optimal margin classifiers* (*Boser et al.* [1992]), which leads directly to the *hard-margin* SVM. Given a set of labeled data, $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where $\mathcal{S}$ is linearly separable, the optimal margin classifier for $\mathcal{S}$ is the hyperplane that is maximally distant from the nearest positive and negative examples. An example of an optimal margin classifier is shown in Figure 3.1. The figure shows two classes—a green class and a yellow class—along with a hyperplane that is equidistant from the nearest green and yellow examples. The minimum distance from the hyperplane to the nearest example represents what we refer to as the geometrical *margin* of the dataset (for simplicity, we will simply refer to this quantity as the *margin* throughout the following sections).

3.2 The Hard-Margin SVM

In this section, we describe a framework for obtaining the optimal margin classifier for a given dataset. This framework leads directly to the the so-called *hard-margin* SVM—the original SVM formulation derived by Vladimir Vapnik and his colleagues. We begin by defining the affine function f:

$$f(\mathbf{x}) \triangleq \mathbf{w}^T\mathbf{x} + b.$$

We refer to f as the *scoring function* of the SVM. Here, $\mathbf{x}$ represents an input feature vector, $\mathbf{w}$ represents a weight vector, and b represents a bias term. The parameters of the SVM are given by $\mathbf{w}$ and b. Given the scoring function f, we use the set $\{\mathbf{x} \ : \ f(\mathbf{x}) = 0\}$ to define a separating hyperplane for performing classification. This is equivalent to using the decision rule in equation (2.1) to arrive at classification hypotheses.

Given f and given a set of labeled training data, $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where $y_i \in \{-1, 1\}$ represents a binary class label, we use the term $\rho_i(\mathbf{w}, b)$ to represent the

geometrical margin of example $\mathbf{x}_i$ given parameters $(\mathbf{w}, b)$. This is defined as follows:

$$\rho_i(\mathbf{w}, b) \triangleq \frac{y_i(\mathbf{w}^T\mathbf{x}_i + b)}{||\mathbf{w}||}.$$

If example x_i is correctly classified by the hyperplane $\{\mathbf{x} : f(\mathbf{x}) = 0\}$, then $\rho_i(\mathbf{w}, b)$ simply represents the minimum Euclidian distance from $\mathbf{x}_i$ to the hyperplane. Given $\rho_i(\mathbf{w}, b)$, we define $\rho(\mathbf{w}, b)$ as the minimum *geometrical margin* over all examples in $\mathcal{S}$:

$$\rho(\mathbf{w}, b) \triangleq \min_i \rho_i(\mathbf{w}, b).$$

The above quantity is commonly referred to as the *geometrical margin* of $\mathcal{S}$ (*Schoelkopf and Smola* [2002]).

If the examples in $\mathcal{S}$ are linearly separable, then we can obtain the optimal margin classifier for $\mathcal{S}$ by solving the following optimization problem:

$$\begin{aligned} & \max_{\mathbf{w},b} \quad \rho(\mathbf{w}_i, b), \\ = & \max_{\mathbf{w},b} \min_i \quad \rho_i(\mathbf{w}, b). \end{aligned}$$

We can change the maximization over $\min_i \rho_i(\mathbf{w}, b)$ into a minimization over $\max_i \frac{1}{\rho_i(\mathbf{w},b)}$ to obtain the following equivalent problem:

$$\begin{aligned} & \max_{\mathbf{w},b} \min_i \quad \rho_i(\mathbf{w}, b), \\ = & \min_{\mathbf{w},b} \max_i \quad \frac{||\mathbf{w}||}{y_i(\mathbf{w}^T\mathbf{x}_i + b)}. \end{aligned}$$

The above problem is homogeneous in $\mathbf{w}$ under the constraints, $1 \leq y_i(\mathbf{w}^T\mathbf{x} + b)$ for all i. (These constraints stipulate that $\mathcal{S}$ must be linearly separable, which is one of our assumptions). Thus, we can restate the optimization problem shown above in the following equivalent form:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \mathbf{w}^T\mathbf{w} \\ \textit{subject to} \quad & 1 \leq y_i(\mathbf{w}^T\mathbf{x}_i + b), \quad \forall i. \end{aligned} \tag{3.1}$$

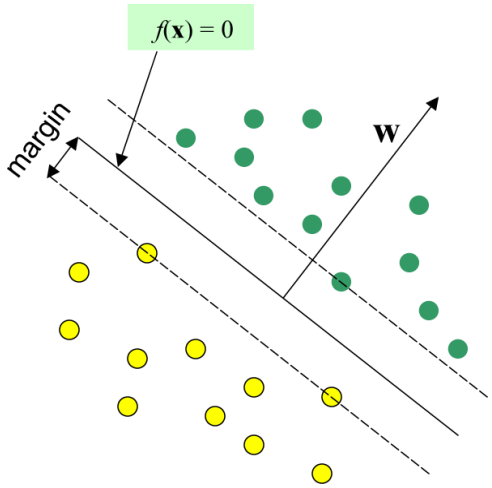

Figure 3.1. Example an *optimal margin classifier* (i.e., a *hard-margin margin SVM*). The decision boundary is represented by the set of points $\{\mathbf{x} : f(\mathbf{x}) = 0\}$.

The above problem forms what we refer to as the *primal problem* of the *hard-margin* SVM. An example of a hard-margin SVM is shown in Figure 3.1. We note that the hard-margin SVM in (3.1) has the form of a specific type of convex optimization problem called a *quadratic program* (QP). Since the problem is convex, we can obtain a global solution to (3.1) by using standard gradient descent techniques. We note however, that in practice, the hard-margin SVM is typically solved by using *interior point methods* for function optimization. Further information on these methods and on other topics in the field of convex optimization can be found in *Boyd and Vandenberghe* [2004].

The primal problem in (3.1) can be converted to an equivalent *dual problem*, which has the following form:

$$\max_{\substack{0 \prec \alpha \\ \alpha^T y = 0}} \quad 2\alpha^T \mathbf{1} - \alpha^T \Lambda_y \mathbf{X}^T \mathbf{X} \Lambda_y \alpha. \tag{3.2}$$

Here, we use $\mathbf{1}$ to represent a column vector of N ones. The terms, y, Λ_y, and $\mathbf{X}$, are

defined as follows:

$$y \triangleq [y_1, \ldots, y_N]^T,$$

$$[\Lambda_y]_{ij} \triangleq \begin{cases} y_i, & \text{if } i = j, \\ 0, & \text{if } i \neq j, \end{cases}$$

$$\mathbf{X} \triangleq [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T.$$

We also define $\alpha \triangleq [\alpha_1, \ldots, \alpha_N]^T$. Here, α_i represents a dual variable, which we use to enforce the constraints, $1 \leq y_i(\mathbf{w}^T\mathbf{x}_i) + b$ for all i. We note that the primal variable $\mathbf{w}$ does not explicitly appear in the dual formulation. However, $\mathbf{w}$ can be obtained from α through the following equation:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i.$$

The above solution for $\mathbf{w}$ follows from the derivation of the dual problem in (3.2). Substituting this expression into the equation for $f(\mathbf{x})$ gives us

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b. \tag{3.3}$$

After solving for the optimal values of α and $\mathbf{w}$, which we represent as α^* and $\mathbf{w}^*$, we can use the constraints of the primal problem in (3.1) to obtain the following solution for the bias term b:

$$b^* = -\frac{1}{2} \cdot \left(\max_{y_i=-1} \mathbf{w}^{*T} \mathbf{x} + \min_{y_i=1} \mathbf{w}^{*T} \mathbf{x} \right)$$

3.2.1 The Kernel Trick

In this section, we examine the so-called "kernel trick" introduced by Vapnik and his colleagues. The kernel trick allows us to implicitly map the $\mathbf{x}$ terms in an SVM from a so-called *input space*, $\mathcal{X}$, into a new, potentially high-dimensional *feature space*, $\mathcal{F}$. This mapping is implemented through a positive semidefinite function called a *kernel*. The key observation behind the kernel trick is that the $\mathbf{x}$ terms only appear in the form of inner products in both the dual problem of (3.2) and in the scoring function of (3.3). Thus, we

can apply the feature mapping $\Phi : \mathbb{R}^L \rightarrow \mathbb{R}^M$ to the input feature vectors by replacing every inner product, $\mathbf{x}_1^T \mathbf{x}_2$, with the kernel function k:

$$k(\mathbf{x}_1, \mathbf{x}_2) \triangleq \Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_2).$$

Here, L represents the dimensionality of the input space $\mathcal{X}$ and M represents the dimensionality of the output feature space $\mathcal{F}$. Any function k that can be expressed in the above form for some feature mapping Φ can be used as a kernel. Equivalently, we say that k is a valid kernel if and only if k forms a positive semidefinite mapping on $\mathbb{R}^L \times \mathbb{R}^L$. This means that we can define valid kernel functions without knowing the exact form of their corresponding feature mappings. We can also define kernels that implicitly map the input space into high-dimensional, and even infinite-dimensional feature spaces. One well-known example of this is the Gaussian kernel, which is defined below:

$$k(\mathbf{x}_1, \mathbf{x}_2) \triangleq exp\Big(- \frac{||\mathbf{x}_1 - \mathbf{x}_2||^2}{\sigma^2} \Big)$$

Here, σ represents the so-called width parameter of k. The Gaussian kernel implicitly maps the input feature vectors into an infinite-dimensional feature space $\mathcal{F}$ using a mapping whose exact form is unknown (at least by me!).

3.2.2 Generalized Linear Kernels

SVMs always train linear or affine decision boundaries in the feature space $\mathcal{F}$ (i.e., the space to which the input feature vectors are mapped by Φ). However, these same decision boundaries can often be highly non-linear when viewed in the original input space $\mathcal{X}$. Thus, one of the main motivations behind using kernels like the Gaussian kernel is that they yield non-linear decision boundaries in the input space, $\mathcal{X}$. This can be useful for datasets where the classes are not linearly separable. However, non-linear decision boundaries typically yield little or no benefit over the standard *linear* or *inner-product* kernel—that is a kernel of the form $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{x}_2$—on tasks where the data *are* linearly separable (or at least approximately separable). We also note that the potential benefits of projecting data into

a high-dimensional or infinite-dimensional space can be dubious when the input feature vectors already have high dimensionality, relative to the number of training examples. For example, in speaker verification, the input feature vectors often have a dimensionality of 20000 or more (this is true of the MLLR-SVM system described in *Stolcke et al.* [2006]; *Hatch et al.* [2006]; however, the total number of positive training examples is typically small (between one and eight), and the total number of negative training examples is also relatively small—typically no more than 5000. Moreover, the distributions of these feature vectors appear to have a high degree of linear separability. For feature sets such as this, we argue that it makes sense to constrain f, and hence the decision boundary defined by $\{\mathbf{x} : f(\mathbf{x}) = 0\}$, to the set of all affine functions in the original input space, $\mathcal{X}$. Based on this argument, we will focus primarily on kernel functions of the following general form:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{R} \mathbf{x}_2.$$

Here, $\mathbf{R}$ represents a square, positive semidefinite parameter matrix. The above kernel can implement any linear feature mapping of the form

$$\Phi(\mathbf{x}) = \mathbf{A}\mathbf{x},$$

where $\mathbf{A}$ is a linear transformation matrix. Note that $\mathbf{R} = \mathbf{A}^T\mathbf{A}$ in this case. In the following chapters, we will refer to functions such as k as *generalized linear kernels*. Chapter 4 provides some background on training kernel functions and kernel parameters, including various techniques for training generalized linear kernels.

3.3 The Soft-Margin SVM

One caveat of the hard-margin SVM is that it can only be applied to datasets that are linearly separable in feature space, $\mathcal{F}$. In practice, one might wish to have an SVM formulation that is guaranteed to yield a decision boundary for *any* choice of dataset—even datasets that are *not* linearly separable in the given feature space. To accomplish this, Vapnik and his colleagues devised what is commonly referred to as the *soft-margin SVM*.

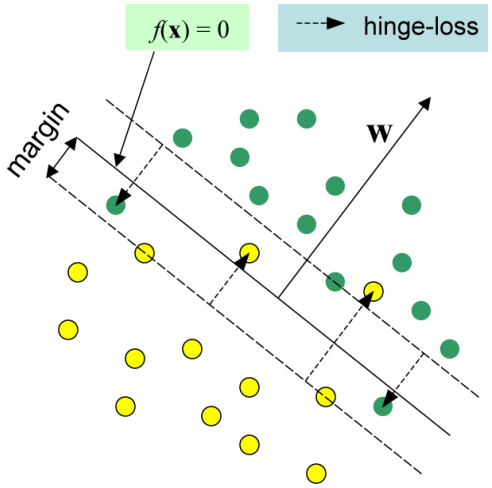

Figure 3.2. Example of a *1-norm soft-margin margin SVM*. The decision boundary is represented by the set of points $\{\mathbf{x} \ : \ f(\mathbf{x}) = 0\}$.

Throughout this dissertation, we will focus specifically on the *1-norm soft margin SVM*, which is perhaps the most well-known and widely-used among the various soft-margin SVM formulations. The primal problem of the 1-norm soft-margin SVM is given below. For brevity, we simply refer to this formulation as the "soft-margin SVM" throughout the remainder of this chapter:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \mathbf{w}^T\mathbf{w} + C\sum_i \xi_i \\ \textit{subject to} \quad & 1 - \xi_i \leq y_i(\mathbf{w}^T\mathbf{x}_i + b), \quad \forall i, \\ & 0 \leq \xi_i \quad \forall i. \end{aligned} \tag{3.4}$$

An example of a 1-norm soft-margin SVM is shown in Figure 3.2. The soft-margin SVM defines a set of *slack variables*, which are represented as $\xi \triangleq [\xi_1, \ldots, \xi_N]$ in the above problem. These variables allow for violations of the margin by relaxing the linear constraints: $1 \leq y_i(\mathbf{w}^T\mathbf{x}_i + b)$ for all i. However, every violation also incurs a penalty (note the $C\sum_i \xi_i$ term in the objective function). From the above optimization problem, we see that each slack variable ξ_i can be represented as

$$\begin{aligned} \xi_i &= (1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))_+, \\ &\geq \mathbf{1}(y_i(\mathbf{w}^T\mathbf{x}_i + b) < 0). \end{aligned}$$

Here, $(x)_+ \triangleq x \cdot \mathbf{1}(x \geq 0)$, and $\mathbf{1}(y_i(\mathbf{w}^T\mathbf{x}_i + b) < 0)$ represents the $0-1$ error function on example $\mathbf{x}_i$. The above relationship shows that ξ_i forms an upper bound on the event that example $\mathbf{x}_i$ is misclassified. We refer to this upper bound as the *hinge-loss* of example $\mathbf{x}_i$. As shown in (3.4) the soft-margin SVM attempts to maximize the margin of the decision boundary by minimizing $\mathbf{w}^T\mathbf{w}$, while also minimizing the total *hinge-loss*, $\sum_i \xi_i$. The tradeoff between maximizing the margin and minimizing hinge-loss is controlled by the C hyperparameter, which is constrained to be positive. In practice, C is often tuned on a cross-validation set. However, various techniques for analytically tuning C have also been proposed (*Cristianini and Shawe-Taylor* [2000]). In Chapters 6 and 7, we derive a new, modified formulation of the 1-norm soft-margin SVM where C is exactly specified.

The optimization problem in (3.4) has the following dual form:

$$\max_{\substack{0 \preceq \alpha \preceq C \\ \alpha^T y = 0}} \quad 2\alpha^T\mathbf{1} - \alpha^T\Lambda_y\mathbf{X}^T\mathbf{X}\Lambda_y\alpha. \tag{3.5}$$

We note that the above problem has the same form as the hard-margin dual in (3.2), except that the α_i terms are bounded above by C. The corresponding $\mathbf{w}$ vector also has the same form as in the hard-margin SVM. Given $\mathbf{w}$, we can compute the optimal (b, ξ) by solving the following linear program (LP):

$$\begin{aligned} \min_{\xi, b} \quad & \sum_i \xi_i \\ \textit{subject to} \quad & 1 - \xi_i \leq y_i(\mathbf{w}^T\mathbf{x}_i + b) \quad \forall i, \\ & 0 \leq \xi_i \quad \forall i. \end{aligned}$$

The soft-margin SVM will play a pivotal role in the following chapters.

Chapter 4

Related Work in the Field of Kernel Optimization

Our goal, in this dissertation, is to examine the problem of kernel optimization for SVM-based speaker recognition, and more generally, for the problem of performing binary classification in multiclass settings. In this chapter, we provide a brief summary of some of the more notable techniques for performing kernel optimization from the literature. The chapter covers a diverse set of techniques, including techniques that are not typically associated with "kernel optimization." For example, we have included a description of *principal component analysis* and *linear discriminant analysis*—techniques that are typically associated with topics such as *feature selection*, *dimensionality reduction*, and *linear analysis*. More generally, these techniques can be viewed as examples of linear feature transformations. Hence, when applied to feature vectors in an SVM, these techniques represent instantiations of a *generalized linear kernel*—that is, a kernel of the form, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}^T\mathbf{R}\mathbf{x}_2$, where $\mathbf{R}$ is a positive semidefinite parameter matrix. The generalized linear kernel will play a pivotal role throughout this dissertation. We also describe a number of kernel techniques and feature transformations that have been developed specifically for speaker verification. For example, we provide a summary of the nuisance attribute projection (NAP) technique

described in *Solomonoff et al.* [2004, 2005], and the n-gram frequency kernel of *Campbell et al.* [2003]. Among non-linear feature transformations, we cover the *rank-normalization* approach described in *Stolcke et al.* [2005]. In Chapters 6 and 8, we compare many of these techniques with a new kernel approach that we refer to as *within-class covariance normalization* (WCCN). This approach is derived in Chapters 5 and 6.

In this chapter, and throughout the dissertation, we pay particular attention to kernel techniques and feature transformations that attempt to model information about clusters or *classes* that reside within the data. Techniques such as this include linear discriminant analysis (LDA) and the nuisance attribute projection (NAP) technique described in *Solomonoff et al.* [2004, 2005]. We refer to these as *supervised* techniques, because they require that each training example $\mathbf{x}_i$ be associated with a user-defined label, y_i. Here, $y_i \in \{1, \ldots, J\}$ can represent any of J classes. Given this label information, a supervised kernel technique can train kernel functions that discriminate between the various classes. In the case of LDA, NAP, and also the WCCN approach that we derive in Chapters 5 and 6, these approaches often boil down to a single linear feature transformation that can be applied uniformly to the input feature space. In Chapter 7, we show how our WCCN approach can be adapted to the particular way in which the classes are *partitioned*—that is, the assignment of each class to either the *target set* or the *impostor set*. We refer to this adaptive form of WCCN as the *adaptive, multicluster SVM* (AMC-SVM).

We also provide a brief description of the so-called *minimax probability machine* (MPM), a kernelizable learning system developed in *Lanckriet et al.* [2002]. The design of the MPM differs significantly from a conventional SVM. However, the MPM is of specific interest to us, because it incorporates information about the first and second-order statistics of both the target and the impostor classes into the training procedure. This use of per-class statistics makes the MPM remarkably comparable to some of the kernel-based techniques that we develop in Chapters 5 through 8. These techniques use information about clusters in the data to tighten the bounds on classification error in an SVM.

4.1 Multiple-Kernel Learning

One of the most well-known kernel learning techniques of the past few years is the *multiple-kernel learning* paradigm of *Lanckriet et al.* [2004]; *Bach et al.* [2004]. Multiple-kernel learning involves training kernel functions as weighted sums of other kernels:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \sum_i \sigma_i k_i(\mathbf{x}_1, \mathbf{x}_2).$$

Here, k_i represents a pre-defined kernel function, and σ_i represents the corresponding weight parameter for k_i. In *Lanckriet et al.* [2004]; *Bach et al.* [2004], the authors show how to learn k by minimizing the SVM dual problem in (3.5) with respect to σ_i for all i. This problem can be posed as a *semidefinite program* (SDP), which reduces to a *second-order cone program* (SOCP) when the σ_i parameters are constrained to be nonnegative (*Lanckriet et al.* [2004]; *Bach et al.* [2004]). The latter case was recently reformulated in *Sonnenburg et al.* [2005] as a semiinfinite *linear program* (LP). Further information about LPs, SOCPs, SDPs, and other convex optimization problems can be found in *Boyd and Vandenberghe* [2004].

In its most recent instantiations, the multiple kernel learning framework tends to be too slow to learn weights for more than a relatively modest number of kernels and training examples (see *Bach et al.* [2004]; *Sonnenburg et al.* [2005] for the latest performance results). Multiple kernel learning also provides no guidance on the question of how to choose a set of basis kernels—that is, the k_i functions. We also note that the implementations described in *Lanckriet et al.* [2004]; *Bach et al.* [2004]; *Sonnenburg et al.* [2005] are designed for general binary classification settings. These implementations make no attempt to use information about clusters that reside *within* the positive and negative classes to obtain tighter bounds on classification error. In Chapters 5 through 8, we show how these issues are at least partially addressed by two new kernel techniques: *within-class covariance normalization* (WCCN) and the *adaptive, multicluster SVM* (AMC-SVM).

4.2 Hyperkernels

Another well-known kernel optimization technique is the *hyperkernels* approach described in *Ong et al.* [2003], where the usual notion of a kernel is expanded to include a "kernel on kernels" (i.e., a *hyperkernel*). Hyperkernels implicitly perform kernel optimization from within a parameterized family of kernels (for instance, a family of Gaussian kernels of varying width parameter, σ). However, as with multiple kernel learning, the hyperkernels approach does not address the issue of how to choose a family of kernels or how to exploit information about subclusters in the data to obtain tighter error bounds.

4.3 Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA)

In this section, we discuss two classical techniques for performing linear feature selection and dimensionality reduction on an input feature space: *principal component analysis* (PCA) and *linear discriminant analysis* (LDA). PCA and LDA are both implemented by performing linear feature transformations on the input feature space. Hence, when applied to feature vectors in an SVM, these techniques represent instantiations of a *generalized linear kernel*—that is, a kernel of the form, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}^T\mathbf{R}\mathbf{x}_2$, where $\mathbf{R}$ is a positive semidefinite parameter matrix.

Principal component analysis (PCA) is a linear technique for reducing the dimensionality of an input feature space while retaining the maximum amount of signal energy. Given an input space of dimensionality N, the goal in PCA is to obtain an orthonormal linear feature transformation, $f(\mathbf{x}) = \theta^T\mathbf{x}$, where θ is defined as an $N \times P$ matrix with $P < N$, such that θ captures the "directions" of maximum energy in the original feature space. We

obtain θ by solving the following optimization problem:

$$\begin{aligned} \max_{\theta} \quad & trace(\theta^T \mathbf{C} \theta) \\ subject\ to \quad & \theta^T \theta = \mathbf{I}. \end{aligned}$$

Here, θ_i represents the ith column vector of matrix θ, and $\mathbf{C}$ represents the overall covariance matrix of the input feature space. The above problem is solved by $\theta^* = \mathbf{V}_P$, where $\mathbf{V}_P$ represents the column matrix containing the top P eigenvectors of $\mathbf{C}$—that is the eigenvectors with the P-largest corresponding eigenvalues. In practice, $\mathbf{C}$ can be estimated empirically as follows:

$$\hat{\mathbf{C}} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T.$$

Here, $\hat{\mathbf{C}}$ represents the empirical covariance matrix computed from a set of N input training examples. We use $\mathbf{x}_n$ to represent the nth training example and $\bar{\mathbf{x}}$ to represent the overall mean of the data. The PCA approach is independent of any associated set of class labels, $\{y_1, \ldots, y_N\}$, for the training data. Thus, we can view PCA as an *unsupervised* approach for performing dimensionality reduction.

Unlike PCA, where the goal is to find orthogonal directions in feature space that retain maximum signal energy, the goal in *linear discriminant analysis* (LDA) is to find orthogonal directions that are "optimal," in some sense, for discriminating between classes. Here, the "optimality" of a given direction is measured as the ratio of between-class variance to within-class variance. We can compute this ratio as follows:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{C}_B \mathbf{w}}{\mathbf{w}^T \mathbf{C}_W \mathbf{w}}.$$

The quantity $J(\mathbf{w})$ represents the ratio of between-class variance to within-class variance for a given direction, $\mathbf{w}$, in feature space. This quantity is traditionally referred to as the *Rayleigh coefficient* for direction $\mathbf{w}$. Given a feature space composed of J classes, we use $\mathbf{C}_B$ and $\mathbf{C}_W$ to represent the *between-class covariance matrix* and the *within-class covariance*

matrix over all classes. These are defined as follows:

$$\mathbf{C}_W \triangleq \sum_{i=1}^{J} p(i)\mathbf{C}_i,$$

$$\mathbf{C}_B \triangleq \sum_{i=1}^{J} p(i)(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})^T.$$

Here, $\mathbf{C}_i$ and $p(i)$ represent the covariance matrix and the prior probability of the ith class. The terms $\bar{\mathbf{x}}_i$ and $\bar{\mathbf{x}}$ represent the mean of class i and the overall mean of the data.

The goal in LDA can be stated as follows: we would like to find the orthonormal linear feature transformation, $f(\mathbf{x}) = \theta^T\mathbf{x}$, where θ is defined as an $N \times P$ matrix with $P < N$, such that the Rayleigh coefficient of each direction in the resulting feature space is maximized. It can be shown that this problem is equivalent to maximizing the ratio of determinants of $\theta^T\mathbf{C}_B\theta$ to $\theta^T\mathbf{C}_W\theta$:

$$\max_{\theta} \quad \frac{|\theta^T\mathbf{C}_B\theta|}{|\theta^T\mathbf{C}_W\theta|}.$$

Here, the optimal θ has a closed-form solution given by $\theta^* = \mathbf{V}_P$, where $\mathbf{V}_P$ represents the column matrix containing the top P eigenvectors of $\mathbf{C}_B\mathbf{C}_W^{-1}$—that is the eigenvectors with the P-largest corresponding eigenvalues. A detailed discussion of LDA, including a set of proofs for the main results, can be found in *Fukunaga* [1990].

4.4 Adaptive Feature Scaling and Relevence Determination

Other notable techniques for kernel optimization include the techniques described in *Weston et al.* [2000] and *Chapelle et al.* [2002]. In these papers, the authors use a set of generalization bounds as objective functions for optimizing various kernel parameters. For example, in *Chapelle et al.* [2002], the authors use the *radius-margin bound* described in *Vapnik* [1995], along with other bounds, as objective functions for simultaneously optimizing the width parameter of a Gaussian kernel and the SVM hyperparameter C. The authors also use a similar approach to train per-feature scaling factors for a standard, linear SVM. For an

L-dimensional feature space, this boils down to the problem of training $\sigma = [\sigma_1, \ldots, \sigma_L]^T$, where $\sigma_i \geq 0$ for all $i \in \{1, \ldots, L\}$ in the following kernel function:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \Lambda_\sigma \mathbf{x}_2.$$

Here, Λ_σ represents an $L \times L$ diagonal matrix, where $\Lambda_{\sigma_{ii}} = \sigma_i$. The above kernel represents a special case of a generalized linear kernel, where the $\mathbf{R}$ parameter matrix is constrained to be diagonal. The problem of training σ is often referred to as *adaptive feature scaling*, *relevence determination*, or *soft feature selection*. In *Chapelle et al.* [2002], the authors iterate between maximizing the SVM dual problem with respect to α for some fixed value of σ (i.e., the standard SVM problem) and minimizing the given generalization bound—for example, the *radius-margin bound* of *Vapnik* [1995]—with respect to σ for fixed α. The latter minimization is achieved by performing gradient descent with respect to σ on the generalization bound. In *Grandvalet and Canu* [2003], the authors propose a modified approach, where the optimization over σ is incorporated into the SVM dual problem. The result is an optimization procedure that only uses a single objective function. As in *Chapelle et al.* [2002], this approach again leads to a slightly complicated, iterative procedure for obtaining the optimized values of σ. The authors of *Grandvalet and Canu* [2003] note that their approach is related to some successful soft feature-selection techniques, such as *lasso* and *bridge* (*Hastie et al.* [2001]) and Automatic Relevence Determination (ARD) (*Neal* [1996]). Other approaches for performing adaptive feature scaling are described in *Bradley and Mangasarian* [1998]; *Jebara and Jaakkola* [2000].

The adaptive scaling approaches described in *Chapelle et al.* [2002]; *Grandvalet and Canu* [2003] provide a means of training a constrained form of a generalized linear kernel (i.e., the $\mathbf{R}$ parameter matrix is constrained to be diagonal). These approaches are technically *supervised* in the sense that they depend on the partitioning of the data into target and impostor sets. However, as was the case with *multiple kernel learning* and *hyperkernels* these approaches may be somewhat limited by the fact that they do not take information about clusters that reside within the data into account when optimizing k. We also note

that these approaches are generally not convex, and that their minimization relies on gradient descent procedures that can be complicated and inefficient. In Chapters 5 through 8, we derive a framework for training *unconstrained* generalized linear kernels through convex optimization. These techniques are *supervised* in that they use cluster information to obtain bounds on classification error.

4.5 The Minimax Probability Machine

In this section, we briefly describe the so-called *minimax probability machine* (MPM), a kernelizable system for training affine decision boundaries for binary classification tasks (*Lanckriet et al.* [2002]). The decision boundary in an MPM is defined by $\{\mathbf{x} : f(\mathbf{x}) = 0\}$, where f has the same general form as in an SVM:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b.$$

As we will show, the MPM training formulation is different than that of an SVM; thus, the MPM does not directly fit into the SVM theme of this chapter. Nevertheless, the MPM is of specific interest to us, because it incorporates information about the first and second-order statistics of both the target and the impostor classes into the training procedure. This use of per-class statistics makes the MPM remarkably comparable to some of the kernel-based techniques that we develop in Chapters 5 through 8. These techniques use information about clusters in the data to tighten the bounds on classification error in an SVM.

Let $(\bar{\mathbf{x}}_{\mathcal{T}}, \mathbf{C}_{\mathcal{T}})$ represent the mean and covariance matrix of a *target class*, $\mathcal{T}$, and let $(\bar{\mathbf{x}}_{\mathcal{I}}, \mathbf{C}_{\mathcal{I}})$ represent the mean and covariance matrix of an *impostor class*, $\mathcal{I}$. We would like to train a decision boundary defined by $\{\mathbf{x} : f(\mathbf{x}) = 0\}$ to separate these two classes. Given $(\bar{\mathbf{x}}_{\mathcal{T}}, \mathbf{C}_{\mathcal{T}})$ and $(\bar{\mathbf{x}}_{\mathcal{I}}, \mathbf{C}_{\mathcal{I}})$, the MPM trains the affine decision boundary that minimizes the maximum probability of misclassification over all distributions of $\mathbf{x}_{\mathcal{T}}$ and $\mathbf{x}_{\mathcal{I}}$, where

$\mathbf{x}_\mathcal{T} \sim (\bar{\mathbf{x}}_\mathcal{T}, \mathbf{C}_\mathcal{T})$ and $\mathbf{x}_\mathcal{I} \sim (\bar{\mathbf{x}}_\mathcal{I}, \mathbf{C}_\mathcal{I})$. This can be expressed as follows:

$$\begin{aligned} \min_{\alpha, \mathbf{w} \neq 0, b} \quad & \alpha \\ \textit{subject to} \quad & \sup_{\mathbf{x}_\mathcal{T} \sim (\bar{\mathbf{x}}_\mathcal{T}, \mathbf{C}_\mathcal{T})} p(\mathbf{w}^T \mathbf{x}_\mathcal{T} + b \leq 0) \leq \alpha, \\ & \sup_{\mathbf{x}_\mathcal{I} \sim (\bar{\mathbf{x}}_\mathcal{I}, \mathbf{C}_\mathcal{I})} p(\mathbf{w}^T \mathbf{x}_\mathcal{I} + b \geq 0) \leq \alpha. \end{aligned} \tag{4.1}$$

Here, α represents the maximum rate of false-positives or false-negatives over all possible distributions of $\mathbf{x}_\mathcal{T}$ and $\mathbf{x}_\mathcal{I}$ that have the given means and covariance matrices. A formal proof of this bound can be found in *Marshall and Olkin* [1960]. In *Lanckriet et al.* [2002], the authors show that the optimization over $\mathbf{w}$ can be restated as follows:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \sqrt{\mathbf{w}^T \mathbf{C}_\mathcal{T} \mathbf{w}} + \sqrt{\mathbf{w}^T \mathbf{C}_\mathcal{I} \mathbf{w}} \\ \textit{subject to} \quad & \mathbf{w}^T (\bar{\mathbf{x}}_\mathcal{T} - \bar{\mathbf{x}}_\mathcal{I}) = 1. \end{aligned} \tag{4.2}$$

The optimization problem for the MPM is convex and can be computed by solving a second-order cone program (SOCP). Further details on the MPM can be found in *Lanckriet et al.* [2002].

4.6 Kernels and Feature Transformations for Speaker Verification

In this section, we discuss some of the more common kernels and feature transformations used in SVM-based speaker verification. Because the feature sets in SVM-based speaker verification tend to have high dimensionality—the feature sets described in Section 2.4 can have anywhere from 10000 to 100000 (or even more) dimensions—and because these feature sets often allow for a high-degree of linear separability between speakers, most SVM-based speaker verification systems use *generalized linear kernels*—that is, kernels of the form, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{R} \mathbf{x}_2$, where $\mathbf{R}$ is a positive semidefinite parameter matrix. We note that the total number of distinct parameters in a generalized linear is on the order of

$N^2/2$, where N is the dimensionality of the input feature space. Thus, generalized linear kernels tend to offer a high degree of modeling power when used in high-dimensional input spaces. Unfortunately, this modeling power (or *capacity* as it was called in Chapter 2) also increases the risk of overfitting. In Chapters 5 through 8, we show how this risk can be managed by optimizing a set of bounds on classification error with respect to $\mathbf{R}$. This leads to a framework where $\mathbf{R}$ is modeled as the inverse of a positively-weighted sum of L covariance matrices, where L is the total number of classes or "clusters" in the data. By modeling $\mathbf{R}$ in this way, we effectively reduce the number of parameters of $\mathbf{R}$ from $N^2/2$ down to L. In this section, we describe various types of generalized linear kernels that have been successfully applied to speaker verification. We also describe the non-linear *rank-normalization* technique developed in *Stolcke et al.* [2005] for normalizing feature vectors prior to training SVM-based speaker models.

4.6.1 Generalized Linear Discriminant Sequence Kernels

One well-known kernel for performing SVM-based speaker verification is the so-called *generalized linear discriminant sequence kernel* (GLDS) kernel of *Campbell* [2001, 2002]. The GLDS kernel essentially corresponds with the parameterization, $\mathbf{R} = \mathbf{C}^{-1}$, where $\mathbf{C}$ is the overall covariance matrix of the data. This choice of $\mathbf{R}$ is not directly tied to any particular bound on classification error. However, the authors show that this parameterization performs a type of discriminative training on the kernel function, $k(\mathbf{x}_1, \mathbf{x}_2)$, where "positive" kernel entries (i.e., the values of $k(\mathbf{x}_1, \mathbf{x}_2)$ where $\mathbf{x}_1$ and $\mathbf{x}_2$ belong to the same class) are discriminated from the so-called "negative" entries, where $\mathbf{x}_1$ and $\mathbf{x}_2$ belong to different classes. The parameterization $\mathbf{R} = \mathbf{C}^{-1}$ performs what we refer to as linear *covariance normalization*. Another common choice for $\mathbf{R}$ is $\mathbf{R} = diag(\mathbf{C})^{-1}$, where $diag(\mathbf{C})$ represents the diagonal component of $\mathbf{C}$. This parameterization performs what we refer to as *per-feature variance normalization*. In Chapter 6, we compare these parameterizations with our own parameterizations for $\mathbf{R}$ on various speaker verification tasks.

4.6.2 N-gram Frequency Kernels

Another widely-used kernel function in speaker verification is derived in *Campbell et al.* [2003]. The kernel function in *Campbell et al.* [2003] is never actually given a name; hence, we will refer to it as the *n-gram frequency kernel*, since this name captures the main idea behind its intended application. Unlike the GLDS kernel, the n-gram frequency kernel is specifically designed for feature vectors whose entries represent relative frequencies of n-grams. The form of the n-gram frequency kernel is given below:

$$k(A,B) = \sum_{i=1}^{M} \frac{p(d_i|convSide_A)}{\sqrt{p(d_i|bkg)}} \frac{p(d_i|convSide_B)}{\sqrt{p(d_i|bkg)}} \tag{4.3}$$

Here, $p(d_i|convSide_A)$ and $p(d_i|bkg)$ refer to the probability (i.e., relative frequency) of n-gram d_i within conversation side A and within the background model, respectively. The above expression represents a kernelized version of the log-likelihood ratio of A given B, or vice-versa. We can also express the n-gram frequency kernel in the form of a generalized linear kernel, where the feature vector for conversation side A is defined as follows:

$$\mathbf{x}_A \triangleq [p(d_1|convSide_A), \ldots, p(d_N|convSide_A)]^T.$$

Under this interpretation, the $\mathbf{R}$ parameter matrix is diagonal, and $[\mathbf{R}]_{ii} = \frac{1}{\sqrt{p(d_i|bkg)}}$. Thus, the n-gram frequency kernel performs a type of per-feature scaling on the input feature space. Further details on this kernel can be found in *Campbell et al.* [2003]. We note that within the field of speaker verification, n-gram frequency kernels are typically applied to speech units such as phonemes and words obtained from an automatic speech recognition system.

4.6.3 Nuisance Attribute Projection (NAP)

Another widely-used kernel technique in the field of speaker verification is the so-called *nuisance attribute projection* (NAP) approach described in *Solomonoff et al.* [2004, 2005]. In its most commonly-used form, NAP is simply a variation of LDA where the between-class covariance matrix is estimated as the identity matrix, $\mathbf{I}$. Under this assumption, the

P-dimensional LDA transformation matrix θ is equal to $\theta = \mathbf{V}_P$, where the columns of $\mathbf{V}_P$ represent the top P eigenvectors of $\mathbf{C}_W^{-1}$. (Equivalently, $\mathbf{V}_P$ represents the bottom P eigenvectors of $\mathbf{C}_W$.) The NAP approach represents one of the most widely-used techniques within the field of speaker verification for training generalized linear kernels. A more thorough description of NAP can be found in *Solomonoff et al.* [2004, 2005].

4.6.4 Rank-Normalization

Among non-linear feature transformations for SVM-based speaker verification, one of the most well-known and widely-used is the *rank-normalization* technique of *Stolcke et al.* [2005]. Rank-normalization uses the following feature transformation:

$$\Phi(x_n) = \frac{1}{N} \arg\min_{i=1}^{N} |x_n - \S_{n,i}|.$$

Here, x_n is the nth feature in feature vector $\mathbf{x}$, and $\S_n = \{x_{n,1}, \ldots, x_{n,N}\}$ is a sorted list of all instances of x_n in the training data (i.e., $x_{n,1} \leq x_{n,2} \leq \cdots \leq x_{n,N}$). Rank normalization applies a non-linear mapping to the features in the training data so that the resulting features are uniformly distributed over the interval, $[0, 1]$. In many cases, this normalization technique has been shown to yield significant improvements over per-feature variance normalization and over covariance normalization in SVM-based speaker recognition systems (*Stolcke et al.* [2005]).

Chapter 5

Error Bounds for Separable Data: A New Derivation of the Hard-Margin SVM

In this chapter, we develop a new theoretical framework for training what we refer to as *generalized linear kernels*—that is, kernels of the form $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{R} \mathbf{x}_2$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are vectors in the input space, and $\mathbf{R}$ is a positive semidefinite matrix. The theory in this chapter focuses specifically on binary classification tasks, where the positive and negative examples are linearly separable within the input feature space, $\mathcal{X}$. We begin by constructing a so-called *class-independent* upper bound on classification error, where all classes in a given set (i.e., either the target set or the impostor set) are assigned the same *bounding function* on the event of a misclassification. Minimizing this upper bound leads to a learning system whose form is similar to the Minimax Probability Machine (MPM) (*Lanckriet et al.* [2002]) summarized in Section 4.5. The class-independent bound can also be extrapolated to obtain a *class-dependent* upper bound on classification error, where the bounding functions are assigned on a per-class basis. We will show that minimizing the class-dependent upper bound leads to a new, modified formulation of the hard-margin SVM.

This modified formulation prescribes a generalized linear kernel where $\mathbf{R}$ is the inverse of a weighted sum of class covariance matrices.

The material in this chapter is organized as follows: Section 5.1 provides a description of our problem setting. Based on this setting, we construct a set of so-called *class-independent* upper bounds on classification error in Section 5.2 and in Section 5.3. These class-independent bounds are then extrapolated in Section 5.4 to obtain a corresponding class-dependent bound. In Section 5.4, we also show how the class-dependent bound leads to a new formulation of the hard-margin SVM and to an analytical form for the $\mathbf{R}$ parameter matrix in a generalized linear kernel.

5.1 Problem Setting

In our problem setting, we are given a multiclass dataset composed of M disjoint classes, where the classes are partitioned *a priori* into two disjoint sets: a *target set*, $\mathcal{T}$, and an *impostor set*, $\mathcal{I}$. We define $y_i \in \{-1, 1\}$ to be the so-called *set label* for class i. Classes that belong to the target set are assigned a label of 1 and classes that belong to the impostor set are assigned a label of -1. Thus, the target and impostor sets are defined as follows:

$$\mathcal{T} = \{i \in 1, \ldots, M \mid y_i = 1\},$$
$$\mathcal{I} = \{i \in 1, \ldots, M \mid y_i = -1\}.$$

Given $\mathcal{T}$ and $\mathcal{I}$, we would like to train a linear classifier that minimizes some measure of binary classification error on these two sets. To do this, we begin by defining the function, f, to be an affine *scoring function*, which we will use to define a decision boundary between $\mathcal{T}$ and $\mathcal{I}$:

$$f(\mathbf{x}) \triangleq \mathbf{v}^T\mathbf{x} + b.$$

Here, $\mathbf{x} \in \mathbb{R}^N$ represents an input feature vector, $\mathbf{v} \in \mathbb{R}^N$ represents a weight vector, and $b \in \mathbb{R}$ represents a bias term. Note that both $\mathbf{v}$ and b represent trainable parameters. Given f, all test examples where $f(\mathbf{x}) \geq 0$ are classified as belonging to $\mathcal{T}$, and all test

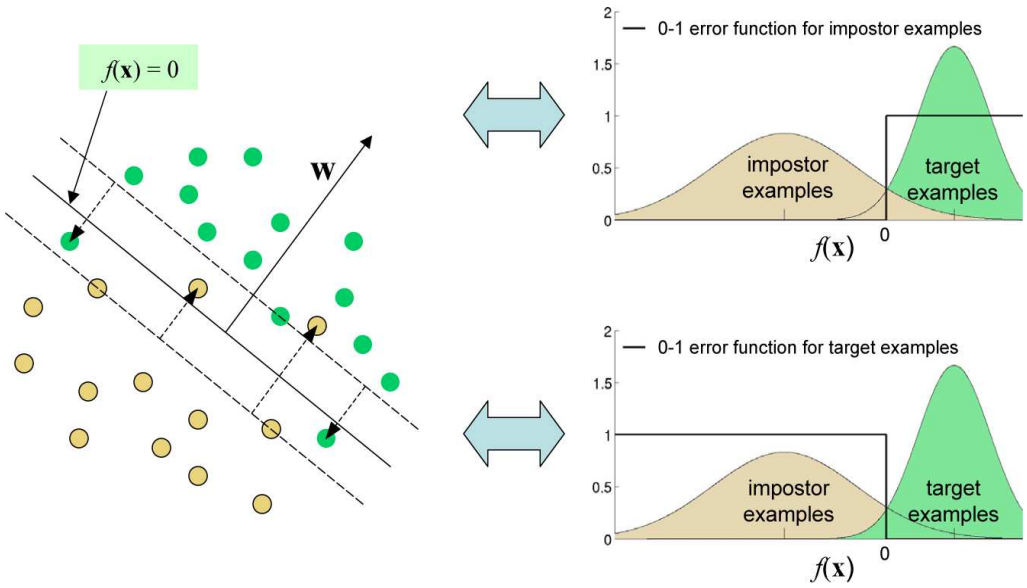

Figure 5.1. Illustration of $0-1$ error functions. The figure on the left shows the decision boundary for a set of *target examples* and for a set of *impostor examples*. The corresponding score distributions and $0-1$ error functions are shown on the right side of the figure.

examples where $f(\mathbf{x}) < 0$ are classified as belonging to $\mathcal{I}$. We can evaluate the classification performance of f by defining the risk metric, $\mathcal{R}(f)$, as

$$\begin{aligned}\mathcal{R}(f) &\triangleq \mathop{\mathbb{E}}_{j\in\mathcal{I}} \mathbf{1}(f(\mathbf{x}_j) \geq 0) + \mathop{\mathbb{E}}_{j\in\mathcal{T}} \mathbf{1}(f(\mathbf{x}_j) < 0), \\ &= p(f(\mathbf{x}_j) \geq 0 \,|\, j \in \mathcal{I}) + p(f(\mathbf{x}_j) < 0 \,|\, j \in \mathcal{T}).\end{aligned}$$

In the above definition, $\mathbf{1}(f(\mathbf{x}) \geq 0)$ represents the so-called $0-1$ *error function* for the impostor examples and $\mathbf{1}(f(\mathbf{x}) < 0)$ represents the $0-1$ *error function* for the target examples. We use the shorthand, $\mathbb{E}_{j\in\mathcal{I}}\,\mathbf{1}(f(\mathbf{x}_j) \geq 0)$, to denote the conditional expectation, $\mathbb{E}(f(\mathbf{x}_j) \geq 0 | j \in \mathcal{I})$. These error functions are illustrated in Figure 5.1, along with the score distributions for a particular target set and impostor set. Taking conditional expectations over these error functions gives us the expected rate of false positives, $p(f(\mathbf{x}_j) \geq 0 \,|\, j \in \mathcal{I})$ and the expected rate of false negatives, $p(f(\mathbf{x}_j) < 0 \,|\, j \in \mathcal{T})$. Our goal is to minimize some upper bound on $\mathcal{R}(f)$ with respect to f—that is, with respect to $\mathbf{v}$ and b.

5.1.1 Notation and Additional Definitions

We use the following notation: Let $\mathbf{x}_i$ be a random draw from class i, and let $\bar{\mathbf{x}}_i$ be the mean of $\mathbf{x}_i$:

$$\bar{\mathbf{x}}_i \triangleq \mathbb{E}\ \mathbf{x}_i.$$

Here, the expectation, $\mathbb{E}\ \mathbf{x}_i$, is taken over all vectors in class i. We define $\mathbf{C}_i$ to be the within-class covariance matrix for class i:

$$\mathbf{C}_j \triangleq \mathbb{E}\,(\mathbf{x}_j - \bar{\mathbf{x}}_j)(\mathbf{x}_j - \bar{\mathbf{x}}_j)^T \quad \forall j.$$

We also define $\mathbf{C}_{\mathcal{T}}$ to be the expected within-class covariance matrix over all classes in the target set and $\mathbf{C}_{\mathcal{I}}$ to be the expected within-class covariance matrix over all classes in the impostor set:

$$\mathbf{C}_{\mathcal{T}} \triangleq \mathop{\mathbb{E}}_{j\in\mathcal{T}} \mathbf{C}_j,$$

$$\mathbf{C}_{\mathcal{I}} \triangleq \mathop{\mathbb{E}}_{j\in\mathcal{I}} \mathbf{C}_j.$$

The overall covariance matrix and the *expected within-class covariance matrix* overall all classes are represented by the symbols, $\mathbf{C}$ and $\mathbf{C}_W$:

$$\mathbf{C} \triangleq \mathop{\mathbb{E}}_{j\in\{\mathcal{T},\mathcal{I}\}} (\mathbf{x}_j - \bar{\mathbf{x}})(\mathbf{x}_j - \bar{\mathbf{x}})^T,$$

$$\mathbf{C}_W \triangleq \mathop{\mathbb{E}}_{j\in\{\mathcal{T},\mathcal{I}\}} \mathbf{C}_j.$$

To simplify our notation in the following sections, we define $\hat{p}_j$ to be the probability of class j conditioned on the given *set* (i.e., either the target set, $\mathcal{T}$, or the impostor set, $\mathcal{I}$):

$$\hat{p}_j \triangleq \begin{cases} \frac{p(j)}{\sum_{k\in\mathcal{T}} p(k)} & \text{if } j \in \mathcal{T}, \\ \frac{p(j)}{\sum_{k\in\mathcal{I}} p(k)} & \text{if } j \in \mathcal{I}. \end{cases}$$

5.2 Bounding Functions

In this section, we construct a set of upper bounds on the risk function, $\mathcal{R}(f)$ for the case where the target class means are linearly separable from the impostor class means

(i.e., $\{\bar{\mathbf{x}}_j\}_{j\in\mathcal{T}}$ is linearly separable from $\{\bar{\mathbf{x}}_j\}_{j\in\mathcal{I}}$). We will use these bounds to derive optimized solutions for $\mathbf{R}$ in the generalized linear kernel, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T\mathbf{R}\mathbf{x}_2$. To simplify these bounds, we assume throughout the following sections that each class is *symmetrically distributed about its mean.* This is formally defined as follows:

Definition 1. *A random variable $\mathbf{x} \in \mathbb{R}^L$ is "symmetrically distributed about its mean" if the following condition holds.*

$$p(\mathbf{x} - \bar{\mathbf{x}} = \Delta) = p(\mathbf{x} - \bar{\mathbf{x}} = -\Delta) \quad \forall(\mathbf{x}, \Delta) \in \mathbb{R}^L \times \mathbb{R}^L.$$

Because $f(\mathbf{x})$ is an affine function of $\mathbf{x}$, one can easily show that if $\mathbf{x}$ is symmetrically distributed about its mean, then $f(\mathbf{x})$ is also symmetrically distributed about its mean. We will use this fact throughout the following chapter to construct upper bounds on classification error for binary decision tasks.

5.3 Class-Independent Bound

We use the setting of Section 5.1 to construct three upper bounds on $\mathcal{R}(f)$ for the case where the target class means are linearly separable from the impostor class means. The first of these bounds is "class-independent," in the sense that the bounding function for a given example, $\mathbf{x}_j$, is the same for all j in set $\mathcal{T}$ and also for all j in $\mathcal{I}$. To derive the upper bound on $\mathcal{R}(f)$, we begin by defining an upper bound on the zero-one loss function, $\mathbf{1}(f(\mathbf{x}_j) > 0)$, for impostor examples.

Theorem 1. *Given the scoring function $f(\mathbf{x}) = \mathbf{v}^T\mathbf{x} + b$, if $f(\bar{\mathbf{x}}_\mathcal{I}) < 0$, then the following inequality holds for all j in $\mathcal{I}$.*

$$\mathbf{1}(f(\mathbf{x}_j) > 0) \leq \left(\frac{f(\mathbf{x}_j) - f(\bar{\mathbf{x}}_\mathcal{I})}{f(\bar{\mathbf{x}}_\mathcal{I})}\right)^2 \cdot \mathbf{1}\big(f(\mathbf{x}_j) > f(\bar{\mathbf{x}}_\mathcal{I})\big) \quad \forall j \in \mathcal{I}. \tag{5.1}$$

Proof. Define $RHS \triangleq \left(\frac{f(\mathbf{x}_j)-f(\bar{\mathbf{x}}_\mathcal{I})}{f(\bar{\mathbf{x}}_\mathcal{I})}\right)^2 \cdot \mathbf{1}\big(f(\mathbf{x}_j) > f(\bar{\mathbf{x}}_\mathcal{I})\big)$ for some j in $\mathcal{I}$. We see that $RHS \geq 0$ for all j. We also see that if $f(\bar{\mathbf{x}}_\mathcal{I}) < 0$, then $RHS = 1$ when $f(\mathbf{x}_j) = 0$, and $RHS \geq 1$ when $f(\mathbf{x}_j) \geq 0$. Thus, we arrive at the inequality in (5.1). □

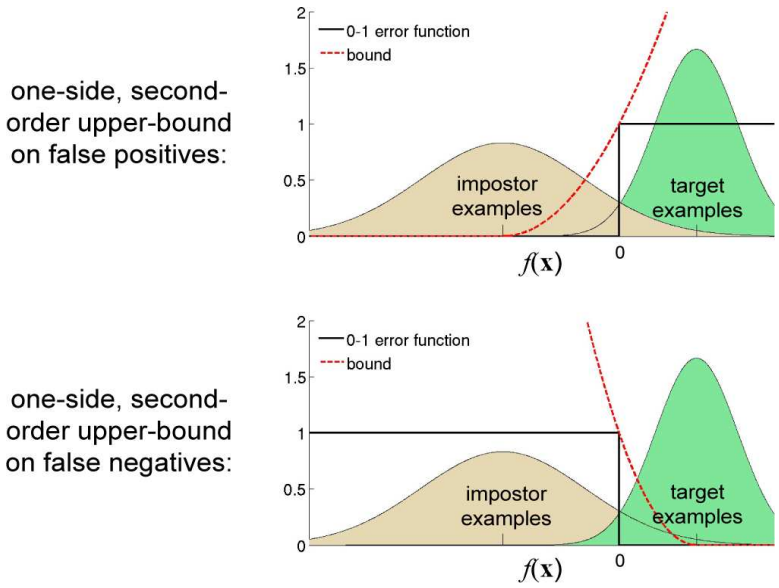

Figure 5.2. Illustration of the class-independent, one-sided, second-order bounding function for the impostor examples and for the target examples.

This bound is illustrated in Figure 5.2. The above inequality defines a one-sided, second-order upper bound on the $0-1$ error function for impostor examples. To simplify the optimization of this bound in the following sections, the second-order bounding function for a given class is centered at the mean of the class. However, we only use the right-hand side of each second-order bounding function; the left-hand side is set to zero (note that the left and right sides are reversed for the corresponding bound on target examples).

We can now use the inequality in (5.1) to obtain the following bound on the risk function, $\mathcal{R}(f)$:

Theorem 2. *Given the scoring function* $f(\mathbf{x}) = \mathbf{v}^T\mathbf{x} + b$, *if* $f(\bar{\mathbf{x}}_{\mathcal{I}}) < 0$ *and* $f(\bar{\mathbf{x}}_{\mathcal{T}}) > 0$, *and if* $\mathbf{x}_{\mathcal{T}}$ *and* $\mathbf{x}_{\mathcal{I}}$ *are symmetrically distributed about their means, then the following bound holds.*

$$\mathcal{R}(f) \leq \frac{1}{2} \cdot \left(\frac{\mathbf{v}^T\mathbf{C}_{\mathcal{T}}\mathbf{v}}{(\mathbf{v}^T\bar{\mathbf{x}}_{\mathcal{T}} + b)^2} + \frac{\mathbf{v}^T\mathbf{C}_{\mathcal{I}}\mathbf{v}}{(\mathbf{v}^T\bar{\mathbf{x}}_{\mathcal{I}} + b)^2} \right). \tag{5.2}$$

Proof. The above bound follows from computing the expectation of bound (5.1) over all impostor classes. This gives us an upper bound on the rate of false positives, $p(f(\mathbf{x}_j) \geq$

$0 \mid j \in \mathcal{I})$. By symmetry, we can compute a similar upper bound on $p(f(\mathbf{x}_j) \leq 0 \mid j \in \mathcal{T})$, the rate of false negatives. Adding the two bounds gives us the upper bound on $\mathcal{R}(f)$ in (5.2). $\square$

The upper bound in (5.2) can also be derived from the Chebyshev inequality, which is given as follows:

$$p(|f(\mathbf{x}) - \mathbb{E}\, f(\mathbf{x})| \geq t) \leq \frac{\mathbf{Var}(f(\mathbf{x}))}{t^2}.$$

For the case where $f(\mathbf{x})$ is symmetrically distributed about its mean, we can convert the above inequality into the following equivalent, one-sided form:

$$p(f(\mathbf{x}) - \mathbb{E}\, f(\mathbf{x}) \geq t) \leq \frac{1}{2} \cdot \frac{\mathbf{Var}(f(\mathbf{x}))}{t^2}.$$

Now, if we substitute $t = -\mathbb{E}\, f(\mathbf{x})$ into the above inequality and constrain $\mathbf{x}$ to the set of all impostor examples, we arrive at the following expression:

$$\begin{aligned} p(f(\mathbf{x}_j) \geq 0 \mid j \in \mathcal{I}) &\leq \frac{1}{2} \cdot \frac{\mathbf{Var}_{j \in \mathcal{I}}(f(\mathbf{x}_j))}{(\mathbb{E}_{j \in \mathcal{I}}\, f(\mathbf{x}_j))^2}, \\ &= \frac{1}{2} \cdot \frac{\mathbf{v}^T \mathbf{C}_{\mathcal{I}} \mathbf{v}}{(\mathbf{v}^T \bar{\mathbf{x}}_{\mathcal{I}} + b)^2}. \end{aligned}$$

By symmetry, we can obtain a similar bound on $p(f(\mathbf{x}_j) < 0 \mid j \in \mathcal{T})$. Adding these bounds gives us the upper bound on $\mathcal{R}(f)$ in (5.2).

We can now use the upper bound in (5.2) as an objective function for training an "optimized" linear classifier. Our goal is to minimize the bound in (5.2) with respect to $(\mathbf{v}, b)$. This gives us the following optimization problem:

$$\begin{aligned} \min_{\mathbf{v},b} \quad & \frac{1}{2} \cdot \left(\frac{\mathbf{v}^T \mathbf{C}_{\mathcal{T}} \mathbf{v}}{(\mathbf{v}^T \bar{\mathbf{x}}_{\mathcal{T}} + b)^2} + \frac{\mathbf{v}^T \mathbf{C}_{\mathcal{I}} \mathbf{v}}{(\mathbf{v}^T \bar{\mathbf{x}}_{\mathcal{I}} + b)^2} \right) \\ \textit{subject to} \quad & 0 < \mathbf{v}^T \bar{\mathbf{x}}_{\mathcal{T}} + b, \\ & 0 > \mathbf{v}^T \bar{\mathbf{x}}_{\mathcal{I}} + b. \end{aligned} \tag{5.3}$$

Here, we have added linear constraints on $\mathbf{v}^T \bar{\mathbf{x}}_{\mathcal{T}} + b$ and on $\mathbf{v}^T \bar{\mathbf{x}}_{\mathcal{I}} + b$ to enforce the assumption in (5.2) that $f(\mathbf{x}_{\mathcal{I}} < 0)$ and that $f(\mathbf{x}_{\mathcal{T}} \geq 0)$. The objective function in (5.3) is composed of terms of the form, $\frac{\mathbf{v}^T \mathbf{C}_{\mathcal{T}} \mathbf{v}}{(\mathbf{v}^T \bar{\mathbf{x}}_{\mathcal{T}} + b)^2}$. These terms are quadratic in $\mathbf{v}$ in both the numerator and

in the denominator. Thus, we say that the objective function in (5.3) is based on terms that have a *quadratic-over-quadratic* functional form. Note that this form is not convex (*Boyd and Vandenberghe* [2004]). However, we can further bound (5.2) to obtain terms that have a *quadratic-over-linear* form, which *is* convex (*Boyd and Vandenberghe* [2004]). The *quadratic-over-linear* form can be bounded even further to obtain a *quadratic program* (QP). The corresponding optimization problems for these bounds on $\mathcal{R}(f)$ are given below, along with the original optimization problem of (5.3).

Theorem 3. *Given the scoring function $f(\mathbf{x}) = \mathbf{v}^T\mathbf{x} + b$, if $\mathbf{x}_\mathcal{T}$ and $\mathbf{x}_\mathcal{I}$ are symmetrically distributed about their means, then the following bounds hold.*

$$\mathcal{R}(f) \leq \min_{\mathbf{v},b} \quad \frac{1}{2}\cdot\left(\frac{\mathbf{v}^T\mathbf{C}_\mathcal{T}\mathbf{v}}{(\mathbf{v}^T\bar{\mathbf{x}}_\mathcal{T}+b)^2}+\frac{\mathbf{v}^T\mathbf{C}_\mathcal{I}\mathbf{v}}{(\mathbf{v}^T\bar{\mathbf{x}}_\mathcal{I}+b)^2}\right) \tag{5.4}$$
$$\textit{subject to} \quad 0 < \mathbf{v}^T\bar{\mathbf{x}}_\mathcal{T}+b,$$
$$0 > \mathbf{v}^T\bar{\mathbf{x}}_\mathcal{I}+b.$$

$$= \min_{\mathbf{v},b} \quad \frac{1}{2}\cdot\left(\frac{\mathbf{v}^T\mathbf{C}_\mathcal{T}\mathbf{v}}{(\mathbf{v}^T\bar{\mathbf{x}}_\mathcal{T}+b)^2}+\frac{\mathbf{v}^T\mathbf{C}_\mathcal{I}\mathbf{v}}{(\mathbf{v}^T\bar{\mathbf{x}}_\mathcal{I}+b)^2}\right) \tag{5.5}$$
$$\textit{subject to} \quad 1 \leq \mathbf{v}^T\bar{\mathbf{x}}_\mathcal{T}+b,$$
$$-1 \geq \mathbf{v}^T\bar{\mathbf{x}}_\mathcal{I}+b.$$

$$\leq \min_{\mathbf{v},b} \quad \frac{1}{2}\cdot\left(\frac{\mathbf{v}^T\mathbf{C}_\mathcal{T}\mathbf{v}}{\mathbf{v}^T\bar{\mathbf{x}}_\mathcal{T}+b}-\frac{\mathbf{v}^T\mathbf{C}_\mathcal{I}\mathbf{v}}{\mathbf{v}^T\bar{\mathbf{x}}_\mathcal{I}+b}\right) \tag{5.6}$$
$$\textit{subject to} \quad 1 \leq \mathbf{v}^T\bar{\mathbf{x}}_\mathcal{T}+b,$$
$$-1 \geq \mathbf{v}^T\bar{\mathbf{x}}_\mathcal{I}+b.$$

$$\leq \min_{\mathbf{v},b} \quad \frac{1}{2}\cdot\mathbf{v}^T(\mathbf{C}_\mathcal{T}+\mathbf{C}_\mathcal{I})\mathbf{v} \tag{5.7}$$
$$\textit{subject to} \quad 1 \leq \mathbf{v}^T\bar{\mathbf{x}}_\mathcal{T}+b,$$
$$-1 \geq \mathbf{v}^T\bar{\mathbf{x}}_\mathcal{I}+b.$$

Proof. The problem in (5.4) is homogeneous in $(\mathbf{v}, b)$. Thus, we can modify the linear

constraints in (5.4) to obtain (5.5). Given these constraints, we can upper bound (5.5) by replacing $(\mathbf{v}^T\bar{\mathbf{x}}_{\mathcal{T}}+b)^2$ with $(\mathbf{v}^T\bar{\mathbf{x}}_{\mathcal{T}}+b)$ and $(\mathbf{v}^T\bar{\mathbf{x}}_{\mathcal{I}}+b)^2$ with $-(\mathbf{v}^T\bar{\mathbf{x}}_{\mathcal{I}}+b)$. This gives us (5.6). We can further upper bound (5.6) by setting the denominators equal to 1 and -1, respectively, as in (5.7). □

In the above set of bounds, the optimization problem in (5.4) represents the original problem in (see (5.3)). We upper bound the optimization problem in (5.4) by the problem in (5.6), where the objective function is composed of terms of the form, $\frac{\mathbf{v}^T\mathbf{C}_{\mathcal{T}}\mathbf{v}}{\mathbf{v}^T\bar{\mathbf{x}}_{\mathcal{T}}+b}$. These terms have a *quadratic-over-linear* form, which is convex (*Boyd and Vandenberghe* [2004]). Moreover, the overall objective function is a positively-weighted sum of convex terms (note that the term, $-\frac{\mathbf{v}^T\mathbf{C}_{\mathcal{I}}\mathbf{v}}{\mathbf{v}^T\bar{\mathbf{x}}_{\mathcal{I}}+b}$, is convex under the constraint, $-1 \geq \mathbf{v}^T\bar{\mathbf{x}}_{\mathcal{I}}+b$). The overall objective function is therefore also convex, as are the constraints. We refer to the functional form of the objective function as a *sum of quadratic-over-linear* form. Since (5.6) has both convex constraints and a convex objective function, the overall optimization problem is also convex. We will show in Chapter 7 that this *sum of quadratic-over-linear* form can be cast as a *second-order cone program* (SOCP).

The *sum of quadratic-over-linear* form in (5.6) is further bounded by the QP in (5.7). We note that if $(\mathbf{C}_{\mathcal{T}}+\mathbf{C}_{\mathcal{I}})$ is full-rank, then the QP in (5.7) can be converted into a more familiar form by defining the vector $\mathbf{w}$ and the matrix $\mathbf{U}$ as follows:

$$\mathbf{v} \triangleq \mathbf{U}\mathbf{w},$$
$$\mathbf{U}\mathbf{U}^T \triangleq (\mathbf{C}_{\mathcal{T}}+\mathbf{C}_{\mathcal{I}})^{-1}.$$

Substituting $\mathbf{U}\mathbf{w}$ in for $\mathbf{v}$ in (5.7) gives us

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\cdot\mathbf{w}^T\mathbf{w} \\ \textit{subject to} \quad & 1 < \mathbf{w}^T\mathbf{U}^T\bar{\mathbf{x}}_{\mathcal{T}}+b, \\ & -1 > \mathbf{w}^T\mathbf{U}^T\bar{\mathbf{x}}_{\mathcal{I}}+b. \end{aligned} \tag{5.8}$$

The above optimization problem has the same general form as the hard-margin SVM in (3.1), except that the feature vectors $\bar{\mathbf{x}}_{\mathcal{T}}$ and $\bar{\mathbf{x}}_{\mathcal{I}}$ have been replaced with $\mathbf{U}^T\bar{\mathbf{x}}_{\mathcal{T}}$ and $\mathbf{U}^T\bar{\mathbf{x}}_{\mathcal{I}}$.

Thus, the formulation in (5.8) implicitly defines the following kernel function k and corresponding feature transformation Φ:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T (\mathbf{C}_\mathcal{T} + \mathbf{C}_\mathcal{I})^{-1} \mathbf{x}_2,$$
$$\Phi(\mathbf{x}) = \mathbf{U}^T \mathbf{x}.$$

From these equations, we see that k is a generalized linear kernel of the form, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{R} \mathbf{x}_2$, where $\mathbf{R}$ is defined as

$$\mathbf{R} = (\mathbf{C}_\mathcal{T} + \mathbf{C}_\mathcal{I})^{-1}.$$

Thus, we have derived a hard-margin SVM along with a generalized linear kernel k and a corresponding feature transformation Φ that are "optimal" in the sense that they minimize the upper bound on classification error in (5.7). In this case, the upper bound is simply based on a pair of one-sided second-order convex bounding functions—one to bound false positives and another to bound false negatives. In the following sections, we will use a similar approach to derive the optimal generalized linear kernel k and feature transformation Φ for more complicated bounding functions. We note that because (5.7) only has two input feature vectors (i.e., $\mathbf{x}_\mathcal{T}$ and $\mathbf{x}_\mathcal{I}$), the optimal $\mathbf{v}$ in (5.7) can be computed analytically as

$$\mathbf{v}^* \propto (\mathbf{C}_\mathcal{T} + \mathbf{C}_\mathcal{I})^{-1} (\mathbf{x}_\mathcal{T} - \mathbf{x}_\mathcal{I}). \tag{5.9}$$

To see this, we begin with the following constraints from the SVM dual formulation of (3.2):

$$\sum_{j \in \{\mathcal{I}, \mathcal{T}\}} \alpha_j y_j = 0,$$
$$0 \leq \alpha_j \quad \forall j.$$

From these constraints, we obtain, $\mathbf{w} \propto \mathbf{U}^T (\mathbf{x}_\mathcal{T} - \mathbf{x}_\mathcal{I})$. Substituting $\mathbf{w} = \mathbf{U}^{-1} \mathbf{v}$ into this equation gives us the solution for $\mathbf{v}^*$ in (5.9).

5.3.1 Comparison with Minimax Probability Machine

In Chapter 4, we described the Minimax Probability Machine (MPM) developed by *Lanckriet et al.* [2002]. Given $(\bar{\mathbf{x}}_\mathcal{T}, \mathbf{C}_\mathcal{T})$ and $(\bar{\mathbf{x}}_\mathcal{I}, \mathbf{C}_\mathcal{I})$, the MPM trains the affine decision

boundary that minimizes the maximum probability of misclassification over all distributions of $\mathbf{x}_\mathcal{T}$ and $\mathbf{x}_\mathcal{I}$, where $\mathbf{x}_\mathcal{T} \sim (\bar{\mathbf{x}}_\mathcal{T}, \mathbf{C}_\mathcal{T})$ and $\mathbf{x}_\mathcal{I} \sim (\bar{\mathbf{x}}_\mathcal{I}, \mathbf{C}_\mathcal{I})$. This can be expressed as follows:

$$\begin{aligned} \min_{\alpha, \mathbf{v} \neq 0, b} \quad & \alpha \\ \textit{subject to} \quad & \sup_{\mathbf{x}_\mathcal{T} \sim (\bar{\mathbf{x}}_\mathcal{T}, \mathbf{C}_\mathcal{T})} p(\mathbf{v}^T \mathbf{x}_\mathcal{T} + b \leq 0) \leq \alpha, \\ & \sup_{\mathbf{x}_\mathcal{I} \sim (\bar{\mathbf{x}}_\mathcal{I}, \mathbf{C}_\mathcal{I})} p(\mathbf{v}^T \mathbf{x}_\mathcal{I} + b \geq 0) \leq \alpha. \end{aligned} \tag{5.10}$$

Here, α represents the maximum rate of false-positives or false-negatives over all possible distributions of $\mathbf{x}_\mathcal{T}$ and $\mathbf{x}_\mathcal{I}$ that have the given means and covariance matrices. In *Lanckriet et al.* [2002], the authors show that the optimization over $\mathbf{v}$ can be restated as follows:

$$\begin{aligned} \min_{\mathbf{v}} \quad & \sqrt{\mathbf{v}^T \mathbf{C}_\mathcal{T} \mathbf{v}} + \sqrt{\mathbf{v}^T \mathbf{C}_\mathcal{I} \mathbf{v}} \\ \textit{subject to} \quad & \mathbf{v}^T (\bar{\mathbf{x}}_\mathcal{T} - \bar{\mathbf{x}}_\mathcal{I}) = 1. \end{aligned} \tag{5.11}$$

The optimization problem for the MPM is convex and can be computed by solving a second-order cone program (SOCP). We can compare (5.11) with the hard-margin SVM from the preceding section. Note that the following optimization problem is equivalent to the problem in (5.7) for optimizing over $\mathbf{v}$:

$$\begin{aligned} \min_{\mathbf{v}} \quad & \sqrt{\mathbf{v}^T \mathbf{C}_\mathcal{T} \mathbf{v} + \mathbf{v}^T \mathbf{C}_\mathcal{I} \mathbf{v}} \\ \textit{subject to} \quad & \mathbf{v}^T (\bar{\mathbf{x}}_\mathcal{T} - \bar{\mathbf{x}}_\mathcal{I}) = 1. \end{aligned} \tag{5.12}$$

Here, we see that the objective functions for the two approaches are actually very similar. The MPM in (5.11) minimizes a sum of square-roots, while the SVM approach in (5.12) minimizes a square-root of sums. The MPM has a potential advantage over the SVM approach in that it achieves the tightest possible bounds on the maximum probability of error for the given means and covariances matrices. However, as we will show in the following sections, the bounding functions that were used to construct the SVM can also be used to construct class-*dependent* bounds, which are often tighter than the class-*independent* bounds in (5.4), (5.6), and (5.7). So far, there are no techniques in the literature for applying the MPM concept to class-dependent bounds.

5.4 Class-Dependent Bounds

The MPM approach and the SVM approach of the previous section are both based on what we refer to as *class-independent* error bounds. One major caveat of both of these approaches is that they only use the means and the covariance matrices of the target and impostor classes to train a decision boundary. In this section, we argue that tighter bounds on classification error can be achieved by constructing so-called *class-dependent* error bounds, where every class gets its own bounding function. These bounds use the means and covariance matrices of *all* classes within the target and impostor sets to train a decision boundary. We begin with a modified version of the inequality in (5.1). This inequality forms an upper bound on the zero-one loss function, $\mathbf{1}(f(\mathbf{x}_j) > 0)$, for impostor class j.

Theorem 4. *Given the scoring function* $f(\mathbf{x}) = \mathbf{v}^T\mathbf{x} + b$, *if* $f(\bar{\mathbf{x}}_j) < 0$ *for all* j *in* $\mathcal{I}$, *then the following inequality holds.*

$$\mathbf{1}(f(\mathbf{x}_j) > 0) \leq \left(\frac{f(\mathbf{x}_j) - f(\bar{\mathbf{x}}_j)}{f(\bar{\mathbf{x}}_j)}\right)^2 \cdot \mathbf{1}\big(f(\mathbf{x}_j) > f(\bar{\mathbf{x}}_j)\big) \quad \forall j \in \mathcal{I}. \tag{5.13}$$

Proof. The proof follows the same steps as the proof of Theorem 1, except that we replace $\bar{\mathbf{x}}_{\mathcal{I}}$ with $\bar{\mathbf{x}}_j$. □

This bound is illustrated in Figure 5.3 for the case where the impostor set is divided into multiple impostor classes. Unlike the class-independent bound of (5.1), where a single function is used to bound the entire impostor set, the bound in (5.13) assigns a separate one-sided, second-order convex bound to every impostor class. The second-order bounding function for a given class is centered at the mean of the class. However, we only use the right-hand side of each second-order bounding function; the left-hand side is set to zero (note that the left and right sides are reversed for the corresponding bound on target examples).

We use the expected value of the class-dependent bounds in (5.13) over all impostor

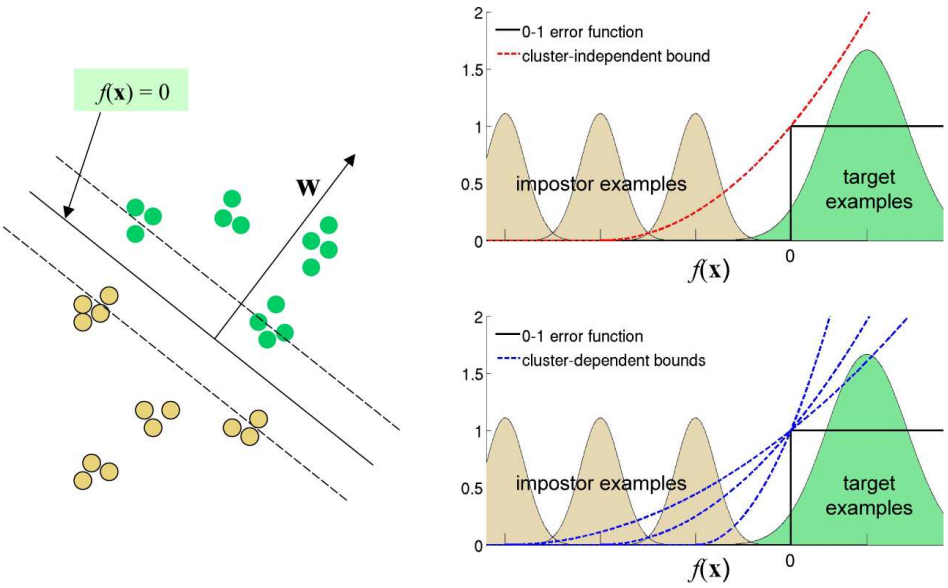

Figure 5.3. Comparison of the class-independent and class-dependent bounding functions for the case where the target and impostor sets are composed of separate classes (i.e., clusters). The figure on the left shows the decision boundary for a set of *target examples* and for a set of *impostor examples*. The corresponding score distributions, $0-1$ error functions, and bounding functions for the impostor examples are shown on the right side of the figure. For simplicity, the score distribution of the target examples is shown as a uni-modal distribution.

classes, along with a corresponding bound for false-positives, to obtain an upper bound on $\mathcal{R}(f)$. This bound is given below:

Theorem 5. *Given the scoring function* $f(\mathbf{x}) = \mathbf{v}^T\mathbf{x} + b$, *if* $f(\bar{\mathbf{x}}_j) < 0$ *for all* $j \in \mathcal{I}$, *and* $f(\bar{\mathbf{x}}_j) > 0$ *for all* $j \in \mathcal{T}$, *and if* $\mathbf{x}_j$ *is symmetrically distributed about its mean for all* j, *then the following bound holds.*

$$\mathcal{R}(f) \leq \frac{1}{2} \cdot \sum_{j \in \{\mathcal{T},\mathcal{I}\}} \hat{p}_j \frac{\mathbf{v}^T\mathbf{C}_j\mathbf{v}}{(\mathbf{v}^T\bar{\mathbf{x}}_j + b)^2}. \tag{5.14}$$

Proof. The above bound follows from computing the expectation of bound (5.13) over all impostor classes. This gives us an upper bound on the rate of false positives, $p(f(\mathbf{x}_j) \geq 0 \,|\, j \in \mathcal{I})$. We can compute a similar upper bound on $p(f(\mathbf{x}_j) \leq 0 \,|\, j \in \mathcal{T})$, the rate of false negatives. Adding the two bounds gives us the upper bound on $\mathcal{R}(f)$ in (5.14). □

The objective function for the above bound is a positively-weighted sum of *quadratic-*

over-quadratic terms. This is the same general functional form as in the class-independent bound on $\mathcal{R}(f)$ in (5.2). Thus, (5.14) also leads to a similar set of optimization problems as those in Section 5.3. These optimization problems are given below:

Theorem 6. *Given the scoring function $f(\mathbf{x}) = \mathbf{v}^T\mathbf{x} + b$, if $\mathbf{x}_{\mathcal{T}}$ and $\mathbf{x}_{\mathcal{I}}$ are symmetrically distributed about their means, then the following bounds hold.*

$$\mathcal{R}(f) \leq \min_{\mathbf{v},b} \quad \frac{1}{2} \cdot \sum_{j \in \{\mathcal{T},\mathcal{I}\}} \hat{p}_j \frac{\mathbf{v}^T\mathbf{C}_j\mathbf{v}}{(\mathbf{v}^T\bar{\mathbf{x}}_j + b)^2} \tag{5.15}$$

$$\textit{subject to} \quad 0 < y_j(\mathbf{v}^T\bar{\mathbf{x}}_j + b) \quad \forall j.$$

$$\leq \min_{\mathbf{v},b} \quad \frac{1}{2} \cdot \sum_{j \in \{\mathcal{T},\mathcal{I}\}} \hat{p}_j \frac{\mathbf{v}^T\mathbf{C}_j\mathbf{v}}{\mathbf{v}^T\bar{\mathbf{x}}_j + b} \tag{5.16}$$

$$\textit{subject to} \quad 1 \leq y_j(\mathbf{v}^T\bar{\mathbf{x}}_j + b) \quad \forall j.$$

$$\leq \min_{\mathbf{v},b} \quad \frac{1}{2} \cdot \mathbf{v}^T\Big(\sum_{j \in \{\mathcal{T},\mathcal{I}\}} \hat{p}_j\mathbf{C}_j\Big)\mathbf{v} \tag{5.17}$$

$$\textit{subject to} \quad 1 \leq y_j(\mathbf{v}^T\bar{\mathbf{x}}_j + b) \quad \forall j.$$

Proof. The proof follows the same steps as the proof for Theorem (3). □

As in Section 5.3, minimizing the upper bound on $\mathcal{R}(f)$ leads to three different optimization problems: a *sum of quadratic-over-quadratic* form (5.15), a *sum of quadratic-over-linear* form (5.16), and a QP (5.17). The *sum of quadratic-over-linear* form is convex and will be discussed in greater detail in Chapter 7.

5.4.1 Hard-Margin SVM

We will focus our attention on the QP in (5.17). If $(\sum_{j \in \{\mathcal{T},\mathcal{I}\}} \hat{p}_j\mathbf{C}_j)$ is full-rank, then the QP in (5.17) can be converted into the standard form for a hard-margin SVM. To show

this, we first define the vector $\mathbf{w}$ and the matrix $\mathbf{U}$ as follows:

$$\mathbf{v} \triangleq \mathbf{U}\mathbf{w},$$
$$\mathbf{U}\mathbf{U}^T \triangleq \Big(\sum_{j \in \{\mathcal{T}, \mathcal{I}\}} \hat{p}_j \mathbf{C}_j \Big)^{-1}.$$

Substituting $\mathbf{U}\mathbf{w}$ in for $\mathbf{v}$ in (5.7) gives us

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \cdot \mathbf{w}^T \mathbf{w} \\ \textit{subject to} \quad & 1 \leq y_j (\mathbf{w}^T \mathbf{U}^T \bar{\mathbf{x}}_j + b) \quad \forall j. \end{aligned} \tag{5.18}$$

The above optimization problem has the same form as a hard-margin SVM (see Chapter 3), except that the feature vector $\bar{\mathbf{x}}_j$ has been replaced with $\mathbf{U}^T \bar{\mathbf{x}}_j$ for all j. However, unlike the hard-margin SVM in (5.8) and the MPM in (5.11), both of which have only two linear constraints—one for $\bar{\mathbf{x}}_{\mathcal{T}}$ and one for $\bar{\mathbf{x}}_{\mathcal{I}}$—the SVM in (5.18) has M linear constraints: one for every class in the data. Thus, the SVM in (5.18) has the same general form as the conventional hard-margin SVM in (3.1), except that it implicitly specifies a kernel function k and feature transformation, Φ. The kernel function k and the corresponding feature transformation Φ for this SVM are defined as follows:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \Big(\sum_{j \in \{\mathcal{T}, \mathcal{I}\}} \hat{p}_j \mathbf{C}_j \Big)^{-1} \mathbf{x}_2, \tag{5.19}$$

$$\Phi(\mathbf{x}) = \mathbf{U}^T \mathbf{x}. \tag{5.20}$$

The term $\mathbf{U}$ represents the Cholesky factorization of $\Big(\sum_{j \in \{\mathcal{T}, \mathcal{I}\}} \hat{p}_j \mathbf{C}_j \Big)^{-1}$:

$$\mathbf{U}\mathbf{U}^T \triangleq \Big(\sum_{j \in \{\mathcal{T}, \mathcal{I}\}} \hat{p}_j \mathbf{C}_j \Big)^{-1}.$$

From these equations, we see that k is a generalized linear kernel of the form, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{R} \mathbf{x}_2$, where $\mathbf{R}$ is defined as

$$\mathbf{R} = \Big(\sum_{j \in \{\mathcal{T}, \mathcal{I}\}} \hat{p}_j \mathbf{C}_j \Big)^{-1}.$$

5.4.2 Bounding Functions for the Hard-Margin SVM

As an alternative to the derivation shown above, the hard-margin SVM of (5.17) can be obtained directly from a specific set of class-dependent bounding functions. These bounding functions are given below.

Theorem 7 (Bounding functions for the hard-margin SVM). *Given a scoring function f, if $y_j f(\bar{\mathbf{x}}_j) > 0$ for all $j \in \{1, \ldots, M\}$, then the following inequality holds.*

$$\mathbf{1}(y_j f(\mathbf{x}_j) < 0) \leq \max_{k \in \{1,\ldots,M\}} \left(\frac{f(\mathbf{x}_j) - f(\bar{\mathbf{x}}_j)}{f(\bar{\mathbf{x}}_k)} \right)^2 \cdot \mathbf{1}\big(y_j f(\mathbf{x}_j) < y_j f(\bar{\mathbf{x}}_j)\big) \quad \forall j \in \{1, \ldots, M\}. \tag{5.21}$$

Proof. The above bound follows from the same steps used in (4). □

Minimizing the expected value of the above bounding functions over all j leads to the optimization problem in (5.17) for the hard-margin SVM. The bounding functions in (5.21) are illustrated in Figure 5.4. Unlike the original second-order bounding functions in (5.13), the second-order functions in (5.21) are constrained to be of uniform width for every class. Figure 5.4 shows that the functions are simply shifted versions of one-another. Thus, the bounds on error-rate are quite loose for classes whose mean scores are far from zero. The upside of this looseness is that the resulting solution for the optimized linear classifier f is relatively simple: the optimized linear classifier is a hard-margin SVM with an optimized linear kernel k and corresponding feature transformation Φ, as given in (5.19) and (5.20). Given this solution, we can train an optimized linear classifier by first applying the linear feature transformation Φ to every feature vector and then training a hard-margin SVM. In Chapter 7, we will show how to tighten the bounding functions in (5.21) while still maintaining the convexity of the overall optimization problem. This leads to a new, modified support vector machine that we refer to as the *adaptive, multicluster SVM*.

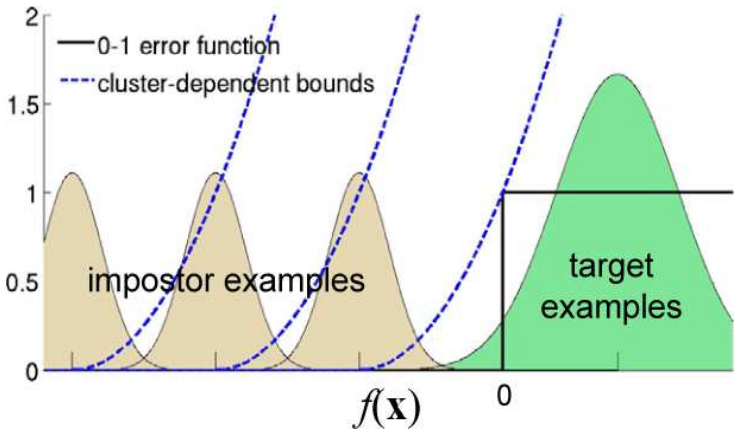

Figure 5.4. A relaxed set of class-dependent bounding functions for the impostor examples.

5.4.3 Bounding Functions for Classes that are not Symmetrically Distributed About their Means

The upper bounds in (5.7) and in (5.17) are somewhat limited by the fact that they only apply to classes that are symmetrically distributed about their means. For non-symmetrical classes, we can use the following class-dependent bounding functions to derive an upper bound on $\mathcal{R}(f)$:

Theorem 8 (Class-dependent bounding functions for asymmetrical classes). *Given the scoring function* $f(\mathbf{x}) = \mathbf{v}^T\mathbf{x} + b$, *if* $y_j f(\bar{\mathbf{x}}_j) > 0$ *for all* $j \in \{1, \ldots, M\}$, *then the following inequality holds.*

$$\mathbf{1}(y_j f(\mathbf{x}_j) < 0) \leq \max_{k \in \{1,\ldots,M\}} \left(\frac{f(\mathbf{x}_j) - f(\bar{\mathbf{x}}_j)}{f(\bar{\mathbf{x}}_k)} \right)^2 \quad \forall j \in \{1, \ldots, M\}. \tag{5.22}$$

Proof. The above bound has the same form as the bound in (7), except that we have removed the indicator function, $\mathbf{1}\big(y_j f(\mathbf{x}_j) < y_j f(\bar{\mathbf{x}}_j)\big)$. Removing the indicator function has the effect of loosening the bound. Thus, the bound in (5.22) is valid. □

The above inequality is the same as the inequality in (5.21), except that we have removed the indicator function, $\mathbf{1}\big(y_j f(\mathbf{x}_j) < y_j f(\bar{\mathbf{x}}_j)\big)$. The result is a two-sided, second-order bounding function. We can obtain the corresponding upper bound on $\mathcal{R}(f)$ by computing

the expected value of (5.22) over all target and impostor examples. Minimizing this bounds gives us the following optimization problem:

Theorem 9 (Class-dependent upper bound on $\mathcal{R}(f)$ for asymmetrical classes). *Given the scoring function $f(\mathbf{x}) = \mathbf{v}^T\mathbf{x} + b$, if $y_j f(\bar{\mathbf{x}}_j) > 0$ for all $j \in \{1, \ldots, M\}$, then the following inequality holds.*

$$\begin{aligned} \mathcal{R}(f) \;\leq\; & \min_{\mathbf{v},b} \quad \mathbf{v}^T\big(\sum_{j\in\{\mathcal{T},\mathcal{I}\}} \hat{p}_j\mathbf{C}_j\big)\mathbf{v} \\ & \textit{subject to} \quad 1 \leq y_j(\mathbf{v}^T\bar{\mathbf{x}}_j + b) \quad \forall j. \end{aligned} \tag{5.23}$$

Proof. The above bound follows from computing the expectation of bound (5.22) over all impostor classes. This gives us an upper bound on the rate of false positives, $p(f(\mathbf{x}_j) \geq 0 \mid j \in \mathcal{I})$. By symmetry, we can compute a similar upper bound on $p(f(\mathbf{x}_j) \leq 0 \mid j \in \mathcal{T})$, the rate of false negatives. Adding the two bounds gives us the upper bound on $\mathcal{R}(f)$ in (5.23). □

Here, we see that (5.23) has the same form as the class-dependent upper bound in (5.17), except that the bound has been multiplied by a factor of 2. A similar bound can be obtained for the class-independent case. The upper bound in (5.23) represents the *worst-case* class-dependent upper bound on $\mathcal{R}(f)$ for any dataset. Although this bound is looser than the bound in (5.17) for the symmetrical case, the two bounds only differ by a constant factor; thus, both bounds yield the same solution for the SVM parameters, $(\mathbf{v}, b)$. The remainder of the dissertation will deal exclusively with upper bounds for classes that are symmetrically distributed about their means. We note, however, that each of these upper bounds can be reformulated for the general case where the class distributions are not assumed to be symmetrical.

5.4.4 Relative Tightness of the Class-Dependent Bounds

We note that the class-dependent bounds of (5.13) and Figure 5.3 do not necessarily lead to a tighter *expected* bound over all classes than the class-independent bound of (5.1)—that is, they do not necessarily lead to tighter bounds on $\mathcal{R}(f)$. For example, if the within-class variance of the scoring function $f(\mathbf{x}_j)$ is relatively large for every class, then the class-dependent bounds will tend to be looser than the class-indepedent bound in expectation over all classes. On the other hand, if the within-class variance of $f(\mathbf{x}_j)$ is small for every class, then the class-dependent bounds may achieve tighter bounds on $\mathcal{R}(f)$ than the class-independent bound. As evidence of this, we can consider the case where the expected within-class variance of $f(\mathbf{x})$ is zero over all classes, but the overall variance of $f(\mathbf{x})$ is σ for some $\sigma > 0$. In this case, the class-independent upper bound on $\mathcal{R}(f)$ in (5.7) will be strictly positive, while the class-dependent bound in (5.17) will be zero.

5.4.5 Clustering Data and Choosing What Constitutes a Class

In general, the relative "tightness" of the class-dependent bounds is dependent on how the classes are defined: classes that are distinct and easily separable from one-another within the data will tend to yield better bounds on $\mathcal{R}(f)$, relative to the class-independent bound, than classes that are indistinct and spread out. Thus, we would obviously prefer that our classes correspond with real *clusters* in the data, where the within-class variance of $f(\mathbf{x}_j)$ over all j is relatively small for any choice of $\mathbf{v}$—that is, for any "direction" within our feature space.

In this thesis, we assume that the classes represent predefined clusters within the target and impostor sets. For example, we will later report results on speaker verification experiments where the classes represent individual speakers. The class-dependent bounds are directly applicable to datasets such as this, where the classes represent real clusters within the data that are defined a priori. In principle, we can also apply the class-dependent bounds to binary classification tasks where the impostor and target sets are *not* partitioned, a pri-

ori, into individual classes. Tasks such as this, are, of course, very common in real world scenarios. In order to apply class-dependent bounds to these general tasks, we must first use clustering techniques to define our own set of clusters (i.e., classes). We will not address the problem of how to perform clustering for class-dependent bounds in any significant depth in this thesis. However, we believe that this problem presents an interesting and important opportunity for future work.

Chapter 6

Error Bounds for Non-Separable Data: A New Derivation of the Soft-Margin SVM

In the preceding chapter, we constructed a set of upper bounds on classification error for the case where the means of the target and impostor classes are linearly separable. By minimizing these upper bounds, we were able to derive a formulation of the hard-margin SVM. This formulation also provides a solution for an optimized linear kernel function k and a corresponding feature transformation, Φ. In this chapter, we extend this approach to the case where the target and impostor means are *not* linearly separable. We describe an approach similar to that of Chapter 5 that leads directly to a new, modified formulation of the soft-margin SVM. This modified formulation differs from the conventional derivation of the soft-margin SVM in *Vapnik* [1995], in the following ways:

1. The new, modified formulation follows directly from minimizing a particular upper bound on classification error. On the other hand, Vapnik's formulation is based on appending *slack variables* to the hard margin SVM.

2. The C hyperparameter is exactly specified in the modified SVM formulation of this chapter but is undetermined in *Vapnik* [1995]. We note, however, that a number of techniques have been proposed for optimizing C analytically (*Cristianini and Shawe-Taylor* [2000]).

3. Our new, modified formulation of the soft-margin SVM provides a solution for an optimized linear kernel k and corresponding feature transformation Φ.

This chapter is organized as follows: We begin by defining a set of bounding functions on the event of a misclassification in Section 6.1. These bounding functions are minimized in Section 6.2 to yield a new formulation of the soft-margin SVM. In Section 6.3, we show how to apply our soft-margin SVM framework to a typical speaker verification task. This section leads to the idea of performing *within-class covariance normalization* (WCCN) on input feature vectors before training SVMs. We discuss the intuition behind WCCN in Section 6.4. Finally, in Sections 6.6 and 6.7, we describe a set of experiments where we compare WCCN to other feature normalizations on a real speaker verification task.

6.1 Bounding Functions

In this section, we construct an upper bound on $\mathcal{R}(f)$ that leads to a modified form of the conventional, 1-norm soft-margin SVM. The upper bound on $\mathcal{R}(f)$ is based on one or more *bounding functions* on the event of a misclassification (i.e., on the $0-1$ error function). As in Chapter 5, bounding functions can be constructed for both the *class-independent* case, where the target and impostor sets are treated as single classes, as well as for the *class-dependent* case, where every class in the target and impostor sets gets its own bounding function. Since the class-indepedent bounding function is a special case of the class-dependent bounding functions, we will assume throughout this chapter that the bounding functions are class-dependent. The bounding functions are defined in Theorem 10. We note that Theorem 10 uses the shorthand $(\cdot)_+$ to represent the *hinge-loss* function.

This is defined as

$$(a)_+ \triangleq \mathbf{1}(a \geq 0) \cdot a.$$

The theorem is given below:

Theorem 10. *Given the scoring function $f(\mathbf{x}) = \mathbf{v}^T\mathbf{x} + b$, if $y_j f(\bar{\mathbf{x}}_j) > 0$ for all classes $j \in \{1, \ldots, M\}$, then the following inequality holds.*

$$\mathbf{1}(y_j f(\mathbf{x}_j) < 0) \leq B(\mathbf{x}_j\,;\,\Theta_j). \tag{6.1}$$

Here, Θ_j represents a set of parameters for class j, and $B(\mathbf{x}_j\,;\,\Theta_j)$ represents a one-sided, second-order bounding function. These are defined as follows:

$$\Theta_j \triangleq \{\mathbf{v}, b, y_j, \bar{\mathbf{x}}_j\},$$
$$B(\mathbf{x}_j\,;\,\Theta_j) \triangleq \big(y_j f(\mathbf{x}_j) - y_j f(\bar{\mathbf{x}}_j)\big)^2\,\mathbf{1}\big(y_j f(\mathbf{x}_j) < y_j f(\bar{\mathbf{x}}_j)\big) + 2 \cdot \big(1 - y_j f(\bar{\mathbf{x}}_j)\big)_+. \tag{6.2}$$

Proof. We can see by inspection that as $y_j f(\mathbf{x}_j)$ decreases, $B(\mathbf{x}_j\,;\,\Theta_j)$ increases or stays the same. Thus, we have only to show that $B(\mathbf{x}_j\,;\,\Theta_j) \geq 1$ for all $y_j f(\bar{\mathbf{x}}_j) \in (0, \infty)$ when $y_j f(\mathbf{x}_j) = 0$. We divide the problem into two cases: one where $y_j f(\bar{\mathbf{x}}_j) \geq 1$, and another where $0 \leq y_j f(\bar{\mathbf{x}}_j) \leq 1$. In each case, we will assume that $y_j f(\mathbf{x}_j) = 0$.

Case 1: $1 \leq y_j f(\bar{\mathbf{x}}_j)$

In this case, the term $2 \cdot \big(1 - y_j f(\bar{\mathbf{x}}_j)\big)_+$ is zero, and the quadratic term $\big(y_j f(\mathbf{x}_j) - y_j f(\bar{\mathbf{x}}_j)\big)^2\,\mathbf{1}\big(y_j f(\mathbf{x}_j) < y_j f(\bar{\mathbf{x}}_j)\big)$ increases as $y_j f(\bar{\mathbf{x}}_j)$ increases. Thus, the bounding function $B(\mathbf{x}_j\,;\,\Theta_j)$ is minimized by setting $y_j f(\bar{\mathbf{x}}_j) = 1$.

Case 2: $0 \leq y_j f(\bar{\mathbf{x}}_j) \leq 1$

For this case, we compute the first and second derivatives of $B(\mathbf{x}_j\,;\,\Theta_j)$ with respect to $y_j f(\bar{\mathbf{x}}_j)$. This gives us

$$\frac{\partial B(\mathbf{x}_j\,;\,\Theta_j)}{\partial\big(y_j f(\bar{\mathbf{x}}_j)\big)} = 2y_j f(\bar{\mathbf{x}}_j) - 2,$$
$$\frac{\partial^2 B(\mathbf{x}_j\,;\,\Theta_j)}{\partial\big(y_j f(\bar{\mathbf{x}}_j)\big)^2} = 2.$$

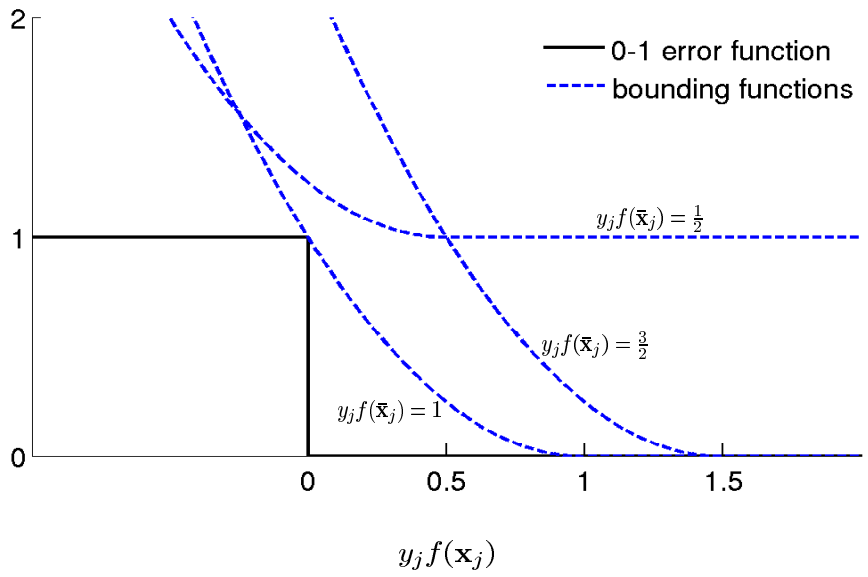

Figure 6.1. Illustration of the $0-1$ error function, $\mathbf{1}(y_j f(\mathbf{x}_j) < 0)$, and the bounding function, $B(\mathbf{x}_j\,;\,\Theta_j)$, as a function of $y_j f(\mathbf{x}_j)$ for various values of $y_j f(\bar{\mathbf{x}}_j)$.

Setting $\frac{\partial B(\mathbf{x}_j\,;\,\Theta_j)}{\partial\left(y_j f(\bar{\mathbf{x}}_j)\right)}$ equal to zero gives us $y_j f(\bar{\mathbf{x}}_j) = 1$. We know that $B(\mathbf{x}_j\,;\,\Theta_j)$ is convex because $\frac{\partial^2 B(\mathbf{x}_j\,;\,\Theta_j)}{\partial\left(y_j f(\bar{\mathbf{x}}_j)\right)^2}$ is strictly positive. Thus, $y_j f(\bar{\mathbf{x}}_j) = 1$ is the global minimum for this case.

The above cases show that $B(\mathbf{x}_j\,;\,\Theta_j)$ is minimized at $y_j f(\bar{\mathbf{x}}_j) = 1$. We also know that $B(\mathbf{x}_j\,;\,\Theta_j) = 1$ at $y_j f(\bar{\mathbf{x}}_j) = 1$. Thus, $B(\mathbf{x}_j\,;\,\Theta_j) \geq 1$ for all $y_j f(\bar{\mathbf{x}}_j) \in (0, \infty)$ when $y_j f(\mathbf{x}_j) = 0$. □

The function, $B(\mathbf{x}_j\,;\,\Theta_j)$, represents a one-sided, second-order bounding function on the $0-1$ error function, $\mathbf{1}(y_j f(\mathbf{x}_j) < 0)$. The latter function equals one when $\mathbf{x}_j$ is misclassified and zero otherwise. An illustration of this bound is provided in Figure 6.1.

6.1.1 Comparison with Bounding Functions for Hard-Margin SVM

We note that the form of $B(\mathbf{x}_j\,;\,\Theta_j)$ in (6.1) is similar to that of the bounding functions for the hard-margin SVM in (5.21). In both (5.21) and (6.1), the bounding functions for the different classes have uniform width and are simply shifted versions of one-another. The

bounding functions in (5.21) have a width equal to $\min_{k\in\{1,\ldots,M\}} y_k f(\bar{\mathbf{x}}_k)$, which represents the maximum possible width for which the bounding functions satisfy the upper bound on $\mathbf{1}(y_j f(\mathbf{x}_j) < 0)$ in (5.21) for all classes. We note, however, that this width assignment only yields valid upper bounds on $\mathbf{1}(y_j f(\mathbf{x}_j) < 0)$ if $\min_{k\in\{1,\ldots,M\}} y_k f(\bar{\mathbf{x}}_k)$ is nonnegative. In other words, the formulation in (5.21) only works in the case where the target and impostor means are linearly separable from one-another given the scoring function, f.

The upper bound in (6.1) is based on the class-dependent bounding function, $B(\mathbf{x}_j\,;\Theta_j)$, which has a uniform width of one for all j. This means that for any class j where $y_j f(\bar{\mathbf{x}}_j) < 1$, the function, $B(\mathbf{x}_j\,;\,\Theta_j)$, will be too wide to upper bound the indicator function in (6.1) in the usual way. To compensate for this, we use the term $\big(1 - y_j f(\bar{\mathbf{x}}_j)\big)_+$ to loosen $B(\mathbf{x}_j\,;\,\Theta_j)$ for any class j where $y_j f(\bar{\mathbf{x}}_j) < 1$. The $\big(1 - y_j f(\bar{\mathbf{x}}_j)\big)_+$ term represents a *hinge-loss* function on $1 - y_j f(\bar{\mathbf{x}}_j)$. In (6.2), this hinge-loss function is scaled by a constant factor of 2. One can show that this is the minimum constant scaling factor for which $B(\mathbf{x}_j\,;\,\Theta_j)$ satisfies the upper bound in (6.1) for any choice of $\bar{\mathbf{x}}_j$.

6.1.2 Upper Bound on $\mathcal{R}(f)$

In this section, we use the inequality in (6.1) to derive an upper bound on $\mathcal{R}(f)$. As was the case in Chapter 5, we assume throughout the following sections that each class is *symmetrically distributed about its mean*. Under this assumption, we can obtain the upper bound on $\mathcal{R}(f)$ given below in Theorem 11. Before giving the theroem, let us define the vector $\xi \triangleq [\xi_1, \ldots, \xi_M]^T$, where M is the number of classes. We will also use the notation, $0 \preceq \xi$, as shorthand to denote a per-element vector inequality:

$$0 \preceq \xi \iff 0 \leq \xi_j \quad \forall j.$$

The theorem is given below:

Theorem 11. *If, for all $j \in \{1, \ldots, M\}$, $\mathbf{x}_j$ is symmetrically distributed about its mean,*

then the following bound holds.

$$\mathcal{R}(f) \leq \frac{1}{2} \cdot \mathbf{v}^T (\sum_j \hat{p}_j \mathbf{C}_j) \mathbf{v} + 2 \cdot \sum_j \hat{p}_j \xi_j \tag{6.3}$$

$$\begin{aligned} \textit{subject to} \quad & 1 - \xi_j \leq y_j (\mathbf{v}^T \bar{\mathbf{x}}_j + b) \quad \forall j, \\ & 0 \preceq \xi. \end{aligned}$$

Proof. A formal proof of Theorem 11 is provided in Appendix A. □

The bound in (6.3) follows from taking the expectation of both sides of the inequality in (6.1).

6.2 The Soft-Margin SVM

If $(\sum_j \hat{p}_j \mathbf{C}_j)$ is full-rank, then we can convert the bound in (6.3) to a more familiar form by performing a substitution of variables. As in Chapter 5, we define the vector $\mathbf{w}$ and the matrix $\mathbf{U}$ as follows:

$$\begin{aligned} \mathbf{v} &\triangleq \mathbf{U}\mathbf{w}, \\ \mathbf{U}\mathbf{U}^T &\triangleq (\sum_j \hat{p}_j \mathbf{C}_j)^{-1}. \end{aligned}$$

Substituting $\mathbf{U}\mathbf{w}$ in for $\mathbf{v}$ in (6.3) gives us the following bound.

$$\mathcal{R}(f) \leq \frac{1}{2} \cdot \mathbf{w}^T \mathbf{w} + 2 \cdot \sum_j \hat{p}_j \xi_j \tag{6.4}$$

$$\begin{aligned} \textit{subject to} \quad & 1 - \xi_j \leq y_j (\mathbf{w}^T \mathbf{U}^T \bar{\mathbf{x}}_j + b) \quad \forall j, \\ & 0 \preceq \xi. \end{aligned}$$

We can now use the upper bound in (6.4) as an objective function for training a linear classifier. Our goal is to minimize the bound in (6.4) with respect to $(\mathbf{w}, b, \xi)$. This gives

us the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \mathbf{w}^T\mathbf{w} + 4 \cdot \sum_j \hat{p}_j \xi_j \\ subject\ to \quad & 1 - \xi_j \leq y_j(\mathbf{w}^T\mathbf{U}^T\bar{\mathbf{x}}_j + b) \quad \forall j, \\ & 0 \preceq \xi. \end{aligned} \tag{6.5}$$

Note that the objective function in (6.5) corresponds with the upper bound in (6.4) scaled by a factor of 2. The optimization problem in (6.5) defines a new, modified formulation of the soft-margin SVM. This modified SVM has the same general form as the conventional 1-norm soft-margin SVM in *Cristianini and Shawe-Taylor* [2000]; *Vapnik* [1995], except that each feature vector, $\bar{\mathbf{x}}_j$, is replaced with $\mathbf{U}^T\bar{\mathbf{x}}_j$. We will discuss the differences between the formulation in (6.5) and the conventional soft-margin SVM formulation in greater detail in the following section. The new SVM formulation implicitly defines a feature mapping Φ and a kernel function k of the form,

$$\Phi(\mathbf{x}) = \mathbf{U}^T\mathbf{x},$$

$$k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T\mathbf{R}\mathbf{x}_2,$$

Here, the matrices, $\mathbf{R}$ and $\mathbf{U}$, are defined as follows:

$$\begin{aligned} \mathbf{R} &\triangleq (\sum_j \hat{p}_j \mathbf{C}_j)^{-1}, \\ \mathbf{U}\mathbf{U}^T &\triangleq \mathbf{R}. \end{aligned} \tag{6.6}$$

6.2.1 Comparison with Conventional Soft-Margin SVM Formulation

If we compare the new, modified soft-margin SVM in (6.5) with the conventional formulation in (3.4), we see that the two formulations are essentially identical. However, there are a few important differences: For example, the new formulation in (6.5) assigns a fixed value of 4 to the SVM hyperparameter, C. This value represents the smallest value of C for which the optimization problem in (6.5) forms an upper bound on the risk function, $\mathcal{R}(f)$. In the conventional soft-margin SVM, the C hyperparameter and the hinge-loss term are

simply appended to the hard-margin SVM without offering any proper justification for their inclusion. Our formulation, on the other hand, follows directly from minimizing the upper bounds on $\mathcal{R}(f)$ in (6.3) and (6.4).

As with the hard-margin SVM in Chapter 5, another key difference between the conventional soft-margin SVM in (3.4) and the new, modified formulation in (6.5) is that the new formulation implicitly specifies a generalized linear kernel k and a corresponding linear feature transformation, Φ (note that this result only holds if $(\sum_j \hat{p}_j \mathbf{C}_j)$ is full-rank). The kernel k and the corresponding feature transformation Φ are selected to minimize the upper bounds on $\mathcal{R}(f)$ in (6.3) and (6.4). Thus, we say that k and Φ are *optimal* for the given upper bounds on classification error. One important implication of this is that the modified SVM in (6.5) is invariant to any full-rank linear distortion of the feature space. This means that if we replace each input feature vector $\bar{\mathbf{x}}_j$ in the training and test sets with $\mathbf{A}\bar{\mathbf{x}}_j$, where $\mathbf{A}$ is some full-rank linear transformation, then the output scores obtained from f will be unchanged. The new formulation in (6.5) also differs from the conventional soft-margin SVM formulation in that it incorporates the prior probabilities of each class conditional on the given set (i.e., either the *target set* or the *impostor set*).

We note that a conventional 1-norm soft-margin SVM with a simple inner-product kernel—that is, a kernel of the form, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{x}_2$—can be viewed as a special case of the optimization problem in (6.5) where the following conditions hold:

1. $\sum_j \hat{p}_j \mathbf{C}_j$ is assumed to be proportional to the identity matrix, $\mathbf{I}$, for all i.
2. The ξ_j terms are weighted by $\hat{p}_j$ for all j
3. Every input training example is treated as the mean of some class. Thus, we replace each $\mathbf{x}_j$ term in the original SVM formulation with $\bar{\mathbf{x}}_j$.
4. The C hyperparameter is set equal to 4.

6.3 Generalized Linear Kernels for Speaker Verification Tasks

In the previous sections, we derived a new, modified formulation of the soft-margin SVM. This SVM specifies an analytical form for $\mathbf{R}$ in the generalized linear kernel, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{R} \mathbf{x}_2$. In this section, we offer a practical argument for how to estimate $\mathbf{R}$ on a typical speaker verification task, where the number of training examples available for the target speaker is very small. Under various assumptions, we show that the formulation in (6.6) leads to the notion of performing *within-class covariance normalization* (WCCN) on the input feature space.

In most speaker verification tasks, the amount of training data provided for the given target speaker is small—typically no more than 8 conversation sides of around 2.5 minutes each. Given this limited amount of training data, the task of coming up with a robust estimate of the covariance matrices for the target set (i.e., speaker) can be very difficult, especially when the dimensionality of the feature space is very high. One way of getting around this, in the absense of any other information, is to assume that the expected within-class covariance matrix over all classes in the target set $\mathcal{T}$ is equal to the expected within-class covariance matrix over *all* classes:

$$\sum_{j \in \mathcal{T}} \hat{p}_j \mathbf{C}_j = \mathbf{C}_W. \tag{6.7}$$

The above assumption will hold—at least approximately—for datasets where the class distributions are selected in such a way that their covariance matrices are relatively homogeneous. We can make a practical argument that the assumption in 6.7 tends to hold for classes that represent instances of a particular *type*. For example, in a speaker verification task, we can define each class to represent an individual *speaker*. We can then use $\mathbf{C}_W$ to provide an estimate of $\mathbf{C}_i$ for any individual speaker i.

Under the assumption in (6.7), the overall $\mathbf{R}$ matrix in (6.6) is approximately equal to

the inverse of the *expected within-class covariance matrix*, $\mathbf{C}_W$:

$$\mathbf{R} \approx \mathbf{C}_W^{-1}.$$

This approximation becomes more exact as the total number of classes, M, grows large. One attractive property of the above assignment for $\mathbf{R}$ is that it applies to any choice of target set, assuming that $\sum_{j \in \mathcal{T}} \hat{p}_j \mathbf{C}_j = \mathbf{C}_W$. Thus, we arrive at a single linear feature mapping that can be applied uniformly to all input feature vectors, regardless of the choice of target set. We can express this feature mapping as follows:

$$\Phi(\mathbf{x}) = \mathbf{C}_W^{-\frac{1}{2}} \mathbf{x}.$$

Here, $\mathbf{C}_W^{\frac{1}{2}}$ represents the Cholesky factorization of $\mathbf{C}_W$. Thus, $\mathbf{C}_W = \mathbf{C}_W^{\frac{1}{2}^T} \mathbf{C}_W^{\frac{1}{2}}$. The above choice of Φ performs what we refer to as *within-class covariance normalization* (WCCN) on the input feature space. This means that Φ normalizes the feature space to have an expected within-class covariance matrix $\mathbf{C}_W$ equal to the identity matrix, $\mathbf{I}$.

6.4 Intuition Behind Within-Class Covariance Normalization

The preceding section shows how the formulation in (6.6) leads to the notion of performing *within-class covariance normalization* (WCCN) on the input feature space. In this section, we describe the intuition behind this approach. Given an input feature space, $\mathcal{X}$, composed of a set of J classes, our goal is to find the linear feature mapping $\Phi(\mathbf{x}) = \mathbf{A}x$ that will optimally transform $\mathcal{X}$ for the purposes of discriminating between classes. More specifically, we would like to find the optimal linear feature mapping for training SVMs in a one-versus-all setting, where one class is chosen as the target class and the remaining $J - 1$ classes are pooled to form the impostor set. If we ignore rotations of the feature space, then the problem of coming up with an optimal linear transformation is equivalent to finding the optimal scaling factor for every direction in feature space. We can argue

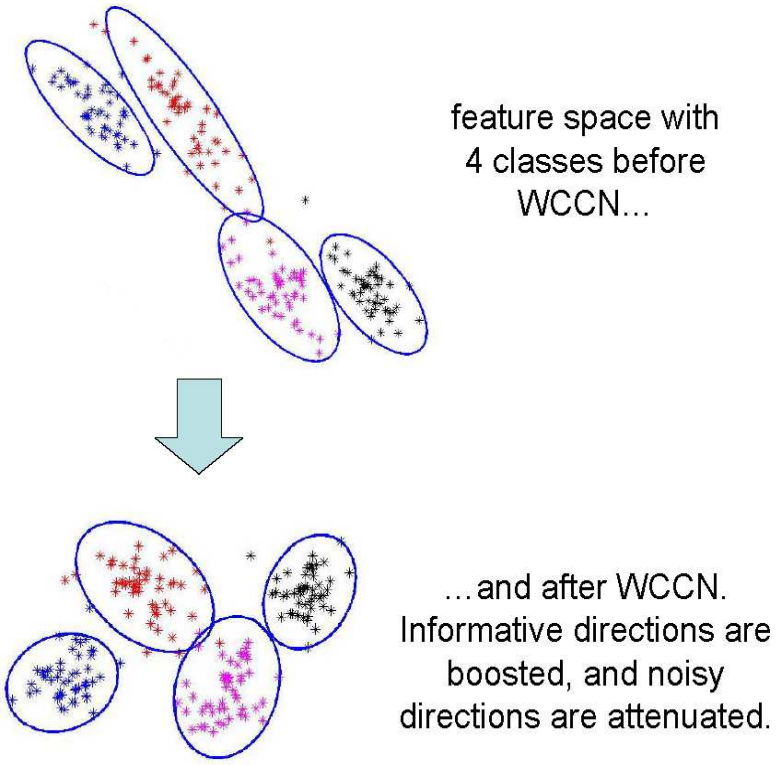

Figure 6.2. Illustration of the *within-class covariance normalization* (WCCN) approach. Here, we apply WCCN to a 2-dimensional feature space composed of 4 Gaussian classes. These ellipses represent isoclines of constant Mahalanobis distance for each of the classes. WCCN boosts informative directions and attenuates noisy directions by making the within-class distributions as symmetrical as possible.

that directions in feature space where the between-class variance is large compared to the expected within-class variance should be given greater weight than directions where this is not the case. Alternatively, we can say that directions with a large Rayleigh coefficient $J(\mathbf{w})$ should be given more weight than directions where $J(\mathbf{w})$ is small. In WCCN, each direction in feature space is scaled by $\frac{1}{\sigma_W(\mathbf{w})}$, where $\sigma_W(\mathbf{w})^2$ represents the expected within class variance over all classes along direction $\mathbf{w}$. After weighting the feature space in this way, each direction will have a between-class variance equal to the Rayleigh coefficient of that direction. Thus, the Rayleigh coefficient forms a measure of how informative or noisy a given direction is. An illustration of how WCCN works is provided in Figure 6.2.

6.5 Relationship Between WCCN and Linear Discriminant Analysis (LDA)

We note that the WCCN approach described in Section 6.3 is, in many respects, very similar to linear discriminant analysis (LDA). In LDA, the input feature space is projected onto the eigenvectors of the matrix $\mathbf{C}_B\mathbf{C}_W^{-1}$. Since the eigenvectors of $\mathbf{C}_B\mathbf{C}_W^{-1}$ are orthonormal, this operation results in a rotation of the input feature space. The resulting feature representation can be truncated—that is, we can drop the features that correspond with the P-lowest Rayleigh coefficients. Beside reducing the dimensionality of the feature space, this truncation can have the effect of filtering out directions in feature space that are noisy. Thus, if P is properly tuned, then LDA will often lead to improved accuracy when used as a pre-processing step in a classification system.

The idea behind WCCN is not to filter out directions in feature space that are noisy, but simply to deemphasize them based on their Rayleigh coefficients. In this way, WCCN attempts to extract as much information as possible from the input feature space $\mathcal{X}$ by optimally weighting every direction. (Note that WCCN is only "optimal" in the sense that it minimizes a particular *upper bound* on classification error). If $\mathbf{C}_W$ is full-rank, then these weights are never equal to zero; thus, WCCN is *lossless*, in that it retains all of the original directions in the input feature space, $\mathcal{X}$. Unlike LDA, WCCN is also invariant to any full-rank linear distortion of $\mathcal{X}$. In other words, we can transform $\mathcal{X}$ with $\Phi(\mathbf{x}) = \mathbf{A}\mathbf{x}$, and the resulting feature space after performing WCCN will look the same for any choice of $\mathbf{A}$ where $\mathbf{A}$ is full-rank and $N \times N$. The same is not true of LDA, however, since LDA simply performs a rotation in feature space followed by a truncation. In Chapter 8, we provide a set of results where we compare the WCCN approach with a version of LDA called *nuisance attribute projection* (NAP) (*Solomonoff et al.* [2004, 2005]). The NAP approach was previously described in Chapter 4.

6.6 Experiments on a Speaker Verification Task

The following section describes a set of experiments where we compare the performance of various linear feature transformations on an SVM-based speaker verification system. These feature transformations include the WCCN approach described in Sections 6.3 through 6.1, where the $\mathbf{R} = \mathbf{C}_W^{-1}$. We also experiment with the parameterization, $\mathbf{R} = \mathbf{C}^{-1}$, where $\mathbf{C}$ is the overall covariance matrix over all of the data. This parameterization corresponds with the *class-independent* case of WCCN where the target and impostor sets are treated as individual classes, and where $\mathbf{C}_{\mathcal{I}}$ and $\mathbf{C}$ are assumed to be equal. We note that this parameterization is identical to the GLDS kernel described in *Campbell* [2001].

Experiments were performed on two NIST-defined speaker verification tasks where the goal is to correctly decide whether or not a given pair of conversation sides belong to the same speaker. In these tasks, one of the conversation sides in each pair is used to define the target class (i.e., speaker), while the other is used as a test example. We train an SVM-based scoring function for every target speaker using a fixed pool of held-out impostor examples taken from several hundred impostor speakers. Note that the classes in these experiments represent speakers.

We used a version of the state-of-the-art MLLR-SVM system described in *Stolcke et al.* [2005] to extract one 12480-dimensional feature vector from every conversation side. These features can be divided into eight disjoint groups of 1560 features each, where each group is associated with a particular set of speech phonemes. We used held-out data from the NIST SRE-2003 dataset to compute the empirical expected within-class covariance matrix $\hat{\mathbf{C}}_W$ and the empirical overall covariance matrix, $\hat{\mathbf{C}}$. Note that both $\hat{\mathbf{C}}_W$ and $\hat{\mathbf{C}}$ were estimated in a block-diagonal fashion, where the covariance between any two features i and j, where i and j belong to different phoneme groups, was set to zero. The resulting covariance matrices

kernel	SRE-03 subset		SRE-04	
	EER%	DCF	EER%	DCF
$\mathbf{R} = diag(\hat{\mathbf{C}}_s)^{-1}$ (baseline)	4.36	0.0166	9.84	0.0347
$\mathbf{R} = diag(\hat{\mathbf{C}}_{W,s})^{-1}$	4.21	0.0151	9.56	0.0338
$\mathbf{R} = \hat{\mathbf{C}}_s^{-1}$	4.15	0.0141	9.56	0.0348
$\mathbf{R} = \hat{\mathbf{C}}_{W,s}^{-1}$	3.80	0.0128	9.28	0.0322
relative improvement	**12.8**%	**22.9**%	**5.7**%	**7.2**%

Table 6.1. EERs and minimum DCFs for various generalized linear kernels. Here, "relative improvement" compares the performance of $\mathbf{R} = \hat{\mathbf{C}}_{W,s}^{-1}$ with the baseline.

were then smoothed using the models,

$$\hat{\mathbf{C}}_{W,s} = \rho_W \cdot \hat{\mathbf{C}}_W + (1 - \rho_W) \cdot diag(\hat{\mathbf{C}}_W), \quad \rho_W \in [0, 1],$$

$$\hat{\mathbf{C}}_s = \rho \cdot \hat{\mathbf{C}} + (1 - \rho) \cdot diag(\hat{\mathbf{C}}), \quad \rho \in [0, 1],$$

where $diag(\mathbf{A})$ is the diagonal component of the square matrix $\mathbf{A}$. The parameters ρ_W and ρ were independently tuned to a value of 0.30 by performing cross-validation on held-out data from the SRE-2003 dataset.

Testing was performed on a subset of the SRE-2003 task and dataset and on the entire SRE-2004 task and dataset for the 1-conversation training condition. Results are shown in Table 6.1 for two standard error metrics: equal-error rate (EER) and minimum *decision cost-function* (DCF)—a standard metric used by NIST to measure classification error when the relative rate of false positives is high (*NIST* [2005]). Note that both error metrics are computed on the pooled set of SVM output scores obtained from the various target classes. Here, the $\mathbf{R} = diag(\hat{\mathbf{C}}_s)^{-1}$ and $\mathbf{R} = diag(\hat{\mathbf{C}}_{W,s})^{-1}$ parameterizations correspond with *per-feature variance normalization* and with *per-feature within-class variance normalization*. The $\mathbf{R} = \hat{\mathbf{C}}_s$ parameterization corresponds with the GLDS kernel (*Campbell* [2001]) and $\mathbf{R} = \hat{\mathbf{C}}_{W,s}$ corresponds with WCCN. As shown in Table 6.1, the $\mathbf{R} = \hat{\mathbf{C}}_{W,s}^{-1}$ case shows a substantial improvement over the $\mathbf{R} = \hat{\mathbf{C}}_s^{-1}$ case and over the baseline, where each feature is normalized to have unit variance (i.e., $\mathbf{R} = diag(\hat{\mathbf{C}})^{-1}$). The improvement on SRE-2004 is significantly smaller than that obtained on SRE-2003. However, this is to be expected, since

both $\hat{\mathbf{C}}_W$ and $\hat{\mathbf{C}}$ were estimated only on SRE-2003 data, which represents a different set of channel and recording conditions than SRE-2004 (*Stolcke et al.* [2005]) for more details about the system, datasets, and tasks). Further information on these experiments can be found in *Hatch and Stolcke* [2006].

6.7 Summary and Conclusions

The preceding chapter describes an approach for training generalized linear kernels of the form, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{R} \mathbf{x}_2$, for OVA classification tasks. We develop a set of error bounds which, under various conditions, are minimized by choosing $\mathbf{R} = \mathbf{C}_W^{-1}$. This parameterization performs what we refer to as *within-class covariance normalization* (WCCN) on the input feature space. In experiments performed on an MLLR-SVM speaker verification system, WCCN achieves substantial reductions in classification error over other linear feature transformations, including the GLDS kernel of *Campbell* [2001] and per-feature variance normalization.

Chapter 7

Tightening the Bounds: the Adaptive, Multicluster SVM

In Chapters 5 and 6, we introduced a new modified formulation of the hard-margin and soft-margin SVMs. These formulations follow from the class-dependent bounding functions in (5.21) and (6.2), which provide an upper bound on the event of a misclassification. The modified SVMs in Chapters 5 and 6 specify a kernel function k and a corresponding linear feature transformation Φ that minimize the upper bounds on $\mathcal{R}(f)$ in (5.18) and (6.4). The upshot of these formulations is that we can implement an "optimized" linear classifier (optimized in the sense that it minimizes these particular upper bounds on $\mathcal{R}(f)$) by performing the following two steps:

1. Use Φ to transform the input feature space.
2. Train a linear SVM in the usual way. Here the term, "linear SVM," refers to an SVM that uses the inner-product kernel, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{x}_2$.

In Section 6.3, we gave a practical argument for how, under certain assumptions, the optimized feature transformation Φ leads to the notion of *within-class covariance normalization* (WCCN). WCCN has the attractive property of being indepedent of the given *partitioning*

of classes—that is, WCCN is independent of the assignment of each class to either the impostor set or to the target set. This means that after performing WCCN on the input feature space, the standard linear SVM is approximately optimal for minimizing the upper bounds in (5.18) and (6.4) on *any* partitioning of classes.

The WCCN approach provides a convenient means of obtaining linear classifiers that approximately minimize a particular upper bound on $\mathcal{R}(f)$. However, WCCN has the drawback of being derived from error-bounds that are unnecessarily loose. We note that the upper bounds on $\mathcal{R}(f)$ in (5.18) and (6.4) are based on a set of one-sided, second-order, class-dependent bounding functions in (5.21) and (6.2) that all have uniform width (see Figure 5.4). These bounding functions are specifically designed to yield objective functions that are easy to solve. However, we can obtain tighter bounds on error by using bounding functions of variable width, where each bounding function intersects the "edge" of the $0-1$ loss function (i.e., the point on the $0-1$ loss function where $f(\mathbf{x}) = 0$). We have already introduced a tighter set of bounding functions in (5.15) and (5.16) for the case where the target and impostor means are linearly separable. These bounding functions are defined to be as wide as possible while still providing an upper bound on the event of a misclassification. In this chapter, we construct a similar set of bounding functions for the case of non-separable data. By minimizing the corresponding upper bound on $\mathcal{R}(f)$, we ultimately arrive at a new formulation of the 1-norm soft-margin SVM that implicitly learns $\mathbf{R}$ in the generalized linear kernel, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{R} \mathbf{x}_2$. Unlike the SVM formulation in Chapter 6, this formulation optimizes the relative weight applied to each class-conditional covariance matrix in the parameter matrix, $\mathbf{R}$. The resulting parameter matrix has the form, $\mathbf{R} = (\sum_j \mu_j \hat{p}_j \mathbf{C}_j)^{-1}$, where μ_j represents the relative weight applied to class j. We refer to this new formulation as the *adaptive, multicluster SVM* (AMC-SVM), because it allows us to adapt the relative weight applied to each class when computing $\mathbf{R}$.

This chapter is organized as follows: Section 7.1 describes a set of bounding functions that form the basis of the AMC-SVM. Section 7.2 describes two methods for optimizing these bounds. These include an *iterated quadratic program* (QP), in which we iterate between

optimizing two different subsets of parameters. We also describe an equivalent formulation, where the AMC-SVM is framed as a second-order cone program (SOCP). In Section 7.3, we describe a set of experiments where we test the AMC-SVM against a conventional SVM that uses WCCN. These experiments are performed on various types of artificial Gaussian data. Finally, a set of conclusions is provided in Section 7.5.

7.1 Bounding Functions

In this section, we construct an upper bound on $\mathcal{R}(f)$. Minimizing this upper bound leads to a modified form of the conventional, 1-norm soft-margin SVM in (3.4). As in Chapter 6, the upper bound on $\mathcal{R}(f)$ follows from a set of class-dependent *bounding functions* on the event of a misclassification (i.e., on the $0-1$ error function). To simplify our notation in the following section, we begin by defining $\mathbf{D}_j$ to be the covariance matrix of class j weighted by the conditional probability of j within the corresponding *set* (i.e., either the target set or the impostor set). Thus, we have:

$$\mathbf{C}_j \triangleq \mathbb{E}\,(\mathbf{x}_j - \bar{\mathbf{x}}_j)(\mathbf{x}_j - \bar{\mathbf{x}}_j)^T \quad \forall j,$$

$$\mathbf{D}_j \triangleq \hat{p}_j \mathbf{C}_j \quad \forall j,$$

where $\hat{p}_j$ is defined as

$$\hat{p}_j \triangleq \begin{cases} \frac{p(j)}{\sum_{k\in\mathcal{T}} p(k)}, & \text{if } \; j \in \mathcal{T}, \\ \frac{p(j)}{\sum_{k\in\mathcal{I}} p(k)}, & \text{if } \; j \in \mathcal{I}. \end{cases}$$

We also define $\bar{\mathbf{X}}$ to be a matrix of the class *means*, and Λ_y to be a diagonal matrix of the corresponding labels:

$$\bar{\mathbf{X}} \triangleq [\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_M],$$

$$[\Lambda_y]_{ij} \triangleq \begin{cases} y_i, & \text{if } \; i = j, \\ 0, & \text{if } \; i \neq j. \end{cases}$$

A set of class-dependent bounding functions is defined below:

Theorem 12. *Given the scoring function* $f(\mathbf{x}) = \mathbf{v}^T\mathbf{x} + b$, *if* $y_j f(\bar{\mathbf{x}}_j) > 0$ *for all* $j \in \{1, \ldots, M\}$, *then the following inequality holds.*

$$\mathbf{1}(y_j f(\mathbf{x}_j) < 0) \leq B(\mathbf{x}_j\,;\,\Theta_j) \quad : \quad 0 < \mu_j \leq 1, \tag{7.1}$$

Here, Θ_j *represents a set of parameters for class* j, *and* $B(\mathbf{x}_j\,;\,\Theta_j)$ *represents a one-sided, second-order bounding function. These are defined as follows:*

$$\Theta_j \triangleq \{\mathbf{v}, b, \mu_j, y_j, \bar{\mathbf{x}}_j\},$$
$$B(\mathbf{x}_j\,;\,\Theta_j) \triangleq \left(\frac{(y_j f(\bar{\mathbf{x}}_j) - y_j f(\mathbf{x}_j))_+}{\max\left\{\frac{1}{\mu_j}, y_j f(\bar{\mathbf{x}}_j)\right\}}\right)^2 + 2\cdot(1 - \mu_j y_j f(\bar{\mathbf{x}}_j))_+. \tag{7.2}$$

Proof. The proof follows essentially the same steps as the proof of Theorem 10. We can see by inspection that as $y_j f(\mathbf{x}_j)$ decreases, $B(\mathbf{x}_j\,;\,\Theta_j)$ increases or stays the same. Thus, we have only to show that $B(\mathbf{x}_j\,;\,\Theta_j) \geq 1$ for all $y_j f(\bar{\mathbf{x}}_j) \in (0, \infty)$ when $y_j f(\mathbf{x}_j) = 0$. We divide the problem into two cases: one where $y_j f(\bar{\mathbf{x}}_j) \geq \frac{1}{\mu_j}$, and another where $0 \leq y_j f(\bar{\mathbf{x}}_j) \leq \frac{1}{\mu_j}$. In each case, we assume that $y_j f(\mathbf{x}_j) = 0$.

Case 1: $\frac{1}{\mu_j} \leq y_j f(\bar{\mathbf{x}}_j)$

In this case, the term $2 \cdot \big(1 - \mu_j y_j f(\bar{\mathbf{x}}_j)\big)_+$ is zero, and the quadratic term $\left(\frac{(y_j f(\bar{\mathbf{x}}_j) - y_j f(\mathbf{x}_j))_+}{\max\left\{\frac{1}{\mu_j}, y_j f(\bar{\mathbf{x}}_j)\right\}}\right)^2$ is equal to one.

Case 2: $0 \leq y_j f(\bar{\mathbf{x}}_j) \leq \frac{1}{\mu_j}$

For this case, we compute the first and second derivatives of $B(\mathbf{x}_j\,;\,\Theta_j)$ with respect to $y_j f(\bar{\mathbf{x}}_j)$. This gives us

$$\frac{\partial B(\mathbf{x}_j\,;\,\Theta_j)}{\partial\big(y_j f(\bar{\mathbf{x}}_j)\big)} = 2\mu_j^2 y_j f(\bar{\mathbf{x}}_j) - 2\mu_j,$$
$$\frac{\partial^2 B(\mathbf{x}_j\,;\,\Theta_j)}{\partial\big(y_j f(\bar{\mathbf{x}}_j)\big)^2} = 2\mu_j^2.$$

Setting $\frac{\partial B(\mathbf{x}_j\,;\,\Theta_j)}{\partial\big(y_j f(\bar{\mathbf{x}}_j)\big)}$ equal to zero gives us $y_j f(\bar{\mathbf{x}}_j) = \frac{1}{\mu_j}$. We know that $B(\mathbf{x}_j\,;\,\Theta_j)$ is convex, because $\frac{\partial^2 B(\mathbf{x}_j\,;\,\Theta_j)}{\partial\big(y_j f(\bar{\mathbf{x}}_j)\big)^2}$ is strictly positive. Thus, $y_j f(\bar{\mathbf{x}}_j) = \frac{1}{\mu_j}$ is the global minimum for this case.

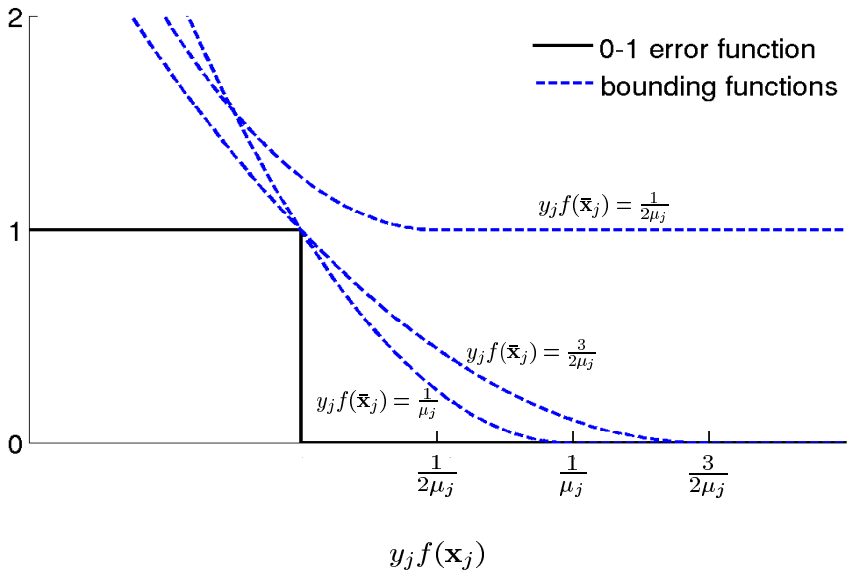

Figure 7.1. Illustration of the $0-1$ error function, $\mathbf{1}(y_j f(\mathbf{x}_j) < 0)$, and the bounding function, $B(\mathbf{x}_j\ ;\ \Theta_j)$, as a function of $y_j f(\mathbf{x}_j)$ for various values of $y_j f(\bar{\mathbf{x}}_j)$. If $\mu_j > 0$, then $B(\mathbf{x}_j\ ;\ \Theta_j)$ forms an upper bound on $\mathbf{1}(y_j f(\mathbf{x}_j) < 0)$ for any value of $y_j f(\mathbf{x}_j)$ and $y_j f(\bar{\mathbf{x}}_j)$.

The above cases show that $B(\mathbf{x}_j\ ;\ \Theta_j)$ is minimized at $y_j f(\bar{\mathbf{x}}_j) = \frac{1}{\mu_j}$. We also know that $B(\mathbf{x}_j\ ;\ \Theta_j) = 1$ at $y_j f(\bar{\mathbf{x}}_j) = \frac{1}{\mu_j}$. Thus, $B(\mathbf{x}_j\ ;\ \Theta_j) \geq 1$ for all $y_j f(\bar{\mathbf{x}}_j) \in (0, \infty)$ when $y_j f(\mathbf{x}_j) = 0$. □

As in Chapter 6, $B(\mathbf{x}_j\ ;\ \Theta_j)$ represents a one-sided, second-order bounding function on the $0-1$ loss function, $\mathbf{1}(y_j f(\mathbf{x}_j) < 0)$. The form of the bounding function in (7.2) is similar to that defined in (6.2), except that we scale $B(\mathbf{x}_j\ ;\ \Theta_j)$ to have a width of $(\max\{\frac{1}{\mu_j}, y_j f(\bar{\mathbf{x}}_j)\})$, where $\mu_j \in (0, 1]$ is a new model parameter. The bounding function, $B(\mathbf{x}_j; \Theta_j)$, is assigned a width of $y_j f(\bar{\mathbf{x}}_j)$ for any class j where $y_j f(\bar{\mathbf{x}}_j) > \frac{1}{\mu_j}$. This represents the maximum possible width for which we satisfy the bound, $\mathbf{1}(y_j f(\mathbf{x}_j) < 0) \leq B(\mathbf{x}_j\ ;\ \Theta_j)$. For the case where $y_j f(\bar{\mathbf{x}}_j) < \frac{1}{\mu_j}$, the bounding function is assigned a fixed width of $\frac{1}{\mu_j}$. We also use the *hinge-loss* term (i.e., $2 \cdot (1 - \mu_j y_j f(\bar{\mathbf{x}}_j))_+$) to ensure that $B(\mathbf{x}_j\ ;\ \Theta_j)$ remains greater than the $0-1$ error function when $y_j f(\bar{\mathbf{x}}_j) < \frac{1}{\mu_j}$. An illustration of $B(\mathbf{x}_j\ ;\ \Theta_j)$ is provided in Figure 7.1. Note that the bound in (7.1) is only valid if μ_j is finite and greater than zero.

We can now use the inequality in (7.1) to derive an upper bound on $\mathcal{R}(f)$. As usual, we assume throughout the following sections that each class is symmetrically distributed about its mean. This implies that for all j, the distribution of $f(\mathbf{x}_j)$ is symmetrical as well, since f is an affine function. Under this assumption, we can obtain the bound in Theorem 13 on the risk function, $\mathcal{R}(f)$. For this bound, we define μ and ξ as $\mu \triangleq [\mu_1, \ldots, \mu_M]^T$, and $\xi \triangleq [\xi_1, \ldots, \xi_M]^T$, where M is the number of classes.

Theorem 13. *If, for all $j \in \{1, \ldots, M\}$, $\mathbf{x}_j$ is symmetrically distributed about its mean, then the following bound holds.*

$$\begin{aligned} \mathcal{R}(f) \leq \frac{1}{2} \cdot \mathbf{v}^T (\sum_j \mu_j \mathbf{D}_j)\mathbf{v} + 2 \cdot \sum_j \hat{p}_j \xi_j & \qquad (7.3) \\ \frac{1}{\mu_j} - \xi_j \leq y_j(\mathbf{v}^T \bar{\mathbf{x}}_j + b) \quad \forall j, & \\ \textit{subject to} \quad 0 \prec \mu \preceq 1, & \\ 0 \preceq \xi. & \end{aligned}$$

Proof. A formal proof of Theorem 13 is provided in *Hatch* [2006] and in Appendix A. □

The above bound follows from taking the expectation of both sides of the inequality in (7.1).

We can convert the bound in (7.3) to a more familiar form by substituting $\mathbf{v} = \mathbf{U}_\mu \mathbf{w}$, where $\mathbf{U}_\mu \mathbf{U}_\mu^T = (\sum_j \mu_j \mathbf{D}_j)^{-1}$. (For simplicity, we will assume throughout this dissertation that $(\sum_j \mu_j \mathbf{D}_j)$ is full-rank and therefore non-singular). This gives us the following bound:

$$\begin{aligned} \mathcal{R}(f) \leq \frac{1}{2} \cdot \mathbf{w}^T \mathbf{w} + 2 \cdot \sum_j \hat{p}_j \xi_j & \qquad (7.4) \\ \frac{1}{\mu_j} - \xi_j \leq y_j(\mathbf{w}^T \mathbf{U}_\mu^T \bar{\mathbf{x}}_j + b) \quad \forall j, & \\ \textit{subject to} \quad 0 \prec \mu \preceq 1, & \\ 0 \preceq \xi. & \end{aligned}$$

Here, we see that for fixed μ, minimizing the bound in (7.4) leads to an optimization problem with the same general form as the 1-norm soft-margin SVM in (3.4), except that the

constraint, $1-\xi_j \leq y_j(\mathbf{w}^T\mathbf{U}_\mu^T\bar{\mathbf{x}}_j+b)\ \forall j$ has been replaced with $\frac{1}{\mu_j}-\xi_j \leq y_j(\mathbf{w}^T\mathbf{U}_\mu^T\bar{\mathbf{x}}_j+b)\ \forall j$. We refer to this formulation, and to affine decision functions that we obtain by minimizing it, as the *adaptive, multicluster SVM* (AMC-SVM). The AMC-SVM implicitly defines a feature mapping Φ and a kernel function k of the form,

$$\Phi(\mathbf{x}) = \mathbf{U}_\mu^T\mathbf{x},$$
$$k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T\mathbf{R}\mathbf{x}_2,$$

where the matrices, $\mathbf{R}$ and $\mathbf{U}_\mu$, are defined as follows:

$$\mathbf{R} \triangleq (\sum_j \mu_j\mathbf{D}_j)^{-1}, \tag{7.5}$$
$$\mathbf{U}_\mu\mathbf{U}_\mu^T \triangleq \mathbf{R}.$$

In the above equations, the μ_j parameters control the relative weight applied to the covariance matrices in computing the feature mapping Φ and the kernel function k.

From the bound in (7.3), we note that a 1-norm soft-margin SVM with a simple inner-product kernel (i.e., a kernel of the form, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T\mathbf{x}_2$) can be viewed as a special case of the AMC-SVM where the following conditions hold:

1. μ_j is fixed at 1 for all j.
2. $\mathbf{C}_j$ is assumed to be proportional to the identity matrix, $\mathbf{I}$, for all j.
3. The ξ_j terms are weighted by $\hat{p}_j$ for all j.
4. Every input training example is treated as the mean of some class. That is, we replace each $\mathbf{x}_j$ term in the original SVM formulation with $\bar{\mathbf{x}}_j$.

Similarly, the WCCN approach of Chapter 6 forms a special case of the AMC-SVM where $\mu_j = 1$ for all j. Hence, we can view the AMC-SVM as an adaptive form of WCCN where the weights assigned to the class covariance matrices (i.e., the $\mathbf{C}_j$ terms) are adapted to the given dataset.

7.2 Optimization

In this section, we examine the problem of minimizing the upper bound on $\mathcal{R}(f)$ in (7.3) with respect to $(\mathbf{v}, b, \xi, \mu)$. We use the notation $(\mathbf{v}^*, b^*, \xi^*, \mu^*)$ to represent the optimizers of (7.3). From (7.3), we see by inspection that μ_j^* has the following solution for all j:

$$\mu_j^* = \frac{1}{y_j(\mathbf{v}^{*T}\bar{\mathbf{x}}_j + b) + \xi_j^*}. \tag{7.6}$$

Given the above solution for μ_j^*, we can state the problem of minimizing (7.3) with respect to $(\mathbf{v}, b, \xi)$ as follows:

$$\begin{aligned} \min_{\mathbf{v},b,\xi} \quad & \sum_j \frac{\mathbf{v}^T\mathbf{D}_j\mathbf{v}}{y_j(\mathbf{v}^T\bar{\mathbf{x}}_j + b) + \xi_j} + 4 \cdot \sum_j \hat{p}_j\xi_j \\ \textit{subject to} \quad & 1 - \xi_j \leq y_j(\mathbf{v}^T\bar{\mathbf{x}}_j + b) \quad \forall j, \\ & 0 \preceq \xi. \end{aligned} \tag{7.7}$$

The $\frac{\mathbf{v}^T\mathbf{D}_j\mathbf{v}}{y_j(\mathbf{v}^T\bar{\mathbf{x}}_j+b)+\xi_j}$ terms in the above optimization problem have a quadratic-over-linear form, which is convex. Thus, the overall objective function of (7.7) is convex, since it is composed of a positively-weighted sum of convex terms. The constraints in (7.7) are linear and therefore also convex; thus, it follows that the overall optimization problem of (7.7) has a convex form.

We prescribe two approaches for solving (7.7). The first of these is an *iterated quadratic program* (QP) approach, where we optimize (7.7) by iteratively solving a QP over $(\mathbf{v}, b, \xi)$ and then minimizing with respect to μ. We also show how the problem in (7.7) can be framed as a second-order cone program (SOCP). This solution was recently proposed by Laurent El Ghaoui.

7.2.1 Iterated QP Formulation

One approach to optimizing (7.7) (or equivalently, to minimizing the bound in (7.3)) is to use what we refer to as an *iterated QP* approach, where we first optimize over $(\mathbf{v}, b, \xi)$ given some initial μ and then over μ given $(\mathbf{v}, b, \xi)$. These two steps can be iterated until

the objective function converges to the global minimum. Although potentially inefficient, this iterative approach has the advantage that it doesn't require any specialized optimization software except for a standard SVM trainer (we note, however, that the iterated QP approach also requires software for performing eigendecompositions on potentially large matrices).

The first step in the iterated QP is to optimize over $(\mathbf{v}, b, \xi)$ given some initial value of μ (say $\mu_j = 1$ for all j). We can express this optimization as a quadratic program (QP):

$$\begin{aligned} \min_{\mathbf{v},b,\xi} \quad & \mathbf{v}^T(\sum_j \mu_j \mathbf{D}_j)\mathbf{v} + C \cdot \sum_j \hat{p}_j \xi_j \\ \textit{subject to} \quad & \tfrac{1}{\mu_j} - \xi_j \leq y_j(\mathbf{v}^T \bar{\mathbf{x}}_j + b) \quad \forall j, \\ & 0 \preceq \xi. \end{aligned} \tag{7.8}$$

$$= \max_{\substack{0 \leq \alpha_j \leq C \cdot \hat{p}_j \ \forall j \\ \alpha^T y = 0}} \quad 2\sum_j \frac{\alpha_j}{\mu_j} - \alpha^T \Lambda_y \bar{\mathbf{X}}^T (\sum_j \mu_j \mathbf{D}_j)^{-1} \bar{\mathbf{X}} \Lambda_y \alpha. \tag{7.9}$$

Here, the optimization problem in (7.9) represents the dual problem of (7.8). We will omit a formal proof of this since the derivation of (7.9) follows essentially the same steps used in *Vapnik* [1995]; *Cristianini and Shawe-Taylor* [2000] to derive the dual of the 1-norm soft-margin SVM. Note that we have replaced the factor "2" in equation (7.3) with the general hyperparameter C in (7.8) and in (7.9). The optimal value of $\mathbf{v}$ for the above optimization problems has the form,

$$\mathbf{v}^* = (\sum_j \mu_j \mathbf{D}_j)^{-1} \bar{\mathbf{X}} \Lambda_y \alpha^*,$$

where α^* is the maximizer of (7.9). This solution for the optimal $\mathbf{v}$ follows from the standard solution in (3.3) for the weight vector of an SVM. Given $\mathbf{v}$ and μ, we can compute the optimal (b, ξ) by solving the following linear program (LP):

$$\begin{aligned} \min_{b,\xi} \quad & \sum_j \hat{p}_j \xi_j \\ \textit{subject to} \quad & \tfrac{1}{\mu_j} - \xi_j \leq y_j(\mathbf{v}^T \bar{\mathbf{x}}_j + b) \quad \forall j, \\ & 0 \leq \xi_j \quad \forall j. \end{aligned}$$

Next, we optimize over μ given $(\mathbf{v}, b, \xi)$:

$$\begin{aligned} \min_{0 \prec \mu \preceq 1} \quad & \mathbf{v}^T (\sum_j \mu_j \mathbf{D}_j) \mathbf{v} + C \cdot \sum_j \hat{p}_j \xi_j \\ \textit{subject to} \quad & \tfrac{1}{\mu_j} - \xi_j \leq y_j (\mathbf{v}^T \bar{\mathbf{x}}_j + b) \quad \forall j. \end{aligned}$$

The solution for the optimal μ in the above optimization problem is given in (7.6). By putting all of these steps together, we arrive at the following iterative procedure for minimizing the bound in (7.3) with respect to $(\mathbf{v}, b, \mu, \xi)$:

1. set $\mu_j := 1 \quad \forall j$.
2. set $\mathbf{R} := (\sum_j \mu_j \mathbf{D}_j)^{-1}$.
3. solve for α:

$$\max_{\substack{0 \leq \alpha_j \leq C \cdot \hat{p}_j \ \forall j \\ \alpha^T y = 0}} \quad 2 \sum_j \frac{\alpha_j}{\mu_j} - \alpha^T \Lambda_y \bar{\mathbf{X}}^T \mathbf{R} \bar{\mathbf{X}} \Lambda_y \alpha. \tag{7.10}$$

4. set $\mathbf{v} := \mathbf{R} \bar{\mathbf{X}} \Lambda_y \alpha$.
5. solve for ξ and b:

$$\begin{aligned} \min_{b, \xi} \quad & \sum_j \hat{p}_j \xi_j \\ \textit{subject to} \quad & \tfrac{1}{\mu_j} - \xi_j \leq y_j (\mathbf{v}^T \bar{\mathbf{x}}_j + b) \quad \forall j, \\ & 0 \leq \xi_j \quad \forall j. \end{aligned}$$

6. set μ:

$$\mu_j := \frac{1}{y_j (\mathbf{v}^T \bar{\mathbf{x}}_j + b) + \xi_j} \quad \forall j.$$

7. return to step 2.

The iterated QP procedure shown above includes a QP in step 3 and an LP in step 5. Both of these problems can be solved by using standard optimization tools for SVMs. The above procedure also requires computing the inverse of a potentially large matrix in step 2. Since matrix inversions can be time-consuming, this step forms one of the main computational

bottlenecks in implementing the iterated QP procedure. As the name implies, the iterated QP also requires that each step be repeated multiple times until the variables converge to the global minimum. Thus, the iterated QP may be somewhat inefficient, particularly in cases where many iterations are required. In the following section, we describe an equivalent formulation of the AMC-SVM where all variables are optimized simultaneously. This formulation has the advantage of being potentially more efficient than the iterated QP; however, it requires an SOCP solver, which may not be readily available to all users.

7.2.2 SOCP Formulation

The optimization problem in (7.7) can also be framed as a second-order cone program (SOCP). To show this, we begin with the following lemma, which was recently proposed by Laurent El Ghaoui:

Lemma 1. *The following inequalities are equivalent for any real scalars $x > 0$ and $y > 0$ and for any vector $\mathbf{z} \in \mathbb{R}^N$, where N is a positive integer:*

$$\begin{aligned} xy &\geq \mathbf{z}^T\mathbf{z} \\ &\iff \frac{1}{2}(x+y) \geq \|\mathbf{z};\ \frac{1}{2}(x-y)\|_2. \end{aligned} \tag{7.11}$$

Proof. Squaring both sides of the inequality in (7.11) gives us

$$\begin{aligned} \frac{1}{2}(x+y) \geq \|\mathbf{z};\ \frac{1}{2}(x-y)\|_2 &\iff \left(\frac{x+y}{2}\right)^2 \geq \|\mathbf{z};\ \frac{x-y}{2}\|_2^2 \\ &\iff \frac{1}{4}(x^2+2xy+y^2) \geq \mathbf{z}^T\mathbf{z} + \frac{1}{4}(x^2-2xy+y^2), \\ &\iff xy \geq \mathbf{z}^T\mathbf{z}. \end{aligned}$$

□

We can use Lemma 1 to obtain the following equivalent SOCP formulation for the bound in (7.7):

Theorem 14 (SOCP Formulation). *The bound on $\mathcal{R}(f)$ in (7.7) can be minimized with respect to $\mathbf{v}$, b, and ξ by solving the following SOCP:*

$$\begin{aligned} \min_{\mathbf{v},b,\xi,t} \quad & \sum_j (t_j + C\hat{p}_j \xi_j) \\ \textit{subject to} \quad & \frac{1}{2}(t_j + y_j(\mathbf{v}^T \bar{\mathbf{x}}_j + b) + \xi_j) \geq \|\mathbf{D}_j^{\frac{1}{2}^T} \mathbf{v};\ \frac{1}{2}(t_j - y_j(\mathbf{v}^T \bar{\mathbf{x}}_j + b) - \xi_j)\|_2 \quad \forall j, \\ & 1 - \xi_j \leq y_j(\mathbf{v}^T \bar{\mathbf{x}}_j + b) \quad \forall j, \\ & 0 \preceq \xi. \end{aligned} \tag{7.12}$$

Here, we define $\mathbf{D}_j^{\frac{1}{2}}$ to be the Cholesky factorization of $\mathbf{D}_j$:

$$\mathbf{D}_j^{\frac{1}{2}} \mathbf{D}_j^{\frac{1}{2}^T} \triangleq \mathbf{D}_j.$$

Proof. The problem in (7.7) can be restated as

$$\begin{aligned} \min_{\mathbf{v},b,\xi,t} \quad & \sum_j (t_j + C\hat{p}_j \xi_j) \\ \textit{subject to} \quad & t_j \geq \frac{\mathbf{v}^T \mathbf{D}_j \mathbf{v}}{y_j(\mathbf{v}^T \bar{\mathbf{x}}_j + b) + \xi_j} \quad \forall j, \\ & 1 - \xi_j \leq y_j(\mathbf{v}^T \bar{\mathbf{x}}_j + b) \quad \forall j, \\ & 0 \preceq \xi. \end{aligned} \tag{7.13}$$

We can now apply Lemma 1 to the linear constraint on t_j. This gives us the SOCP in (7.12). □

Note that in Theorem 14, the term t is defined as $t \triangleq [t_1, \ldots, t_M]^T$, where M is the number of classes.

7.2.3 A Kernelized Version of the Adaptive, Multicluster SVM

In this section, we address the problem of how to empirically estimate class covariance matrices, and more specifically, how to estimate $\mathbf{R} \triangleq (\sum_j \mu_j \mathbf{D}_j)^{-1}$ given a finite set of

training data. The theory in this section also leads to a "kernelized" version of the AMC-SVM, which allows us to apply the iterated QP and SOCP formulations of the previous sections to arbitrary kernels and feature mappings. For instance, the theory developed in this section can be used to implement an AMC-SVM with a Gaussian or RBF kernel. We begin, in the following section, by deriving an iterated QP formulation of the AMC-SVM where the class covariance matrices are estimated empirically. This is followed by a similar derivation for the SOCP formulation.

Iterated QP Formulation

We begin by defining $\mathbf{X}_i$ to be a matrix containing the training vectors for class i:

$$\mathbf{X}_i \triangleq [\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_{N_i}}].$$

Here, $\mathbf{x}_{i_j}$ represents the jth training vector of class i, and N_i represents the total number of training vectors in the class. Next, we define $\hat{\mathbf{X}}_i$ to be a weighted and centered version of $\mathbf{X}_i$:

$$\hat{\mathbf{X}}_i \triangleq \sqrt{\frac{\hat{p}_i}{N_i}}\Big(\mathbf{X}_i - \frac{1}{N_i}\mathbf{X}_i \mathbf{1}_{N_i}\mathbf{1}_{N_i}^T\Big).$$

We use the notation $\mathbf{1}_{N_i}$ to denote a column vector of N_i ones. Given the above definition for $\hat{\mathbf{X}}_i$, we define $\hat{\mathbf{X}}$ to be a matrix containing $\hat{\mathbf{X}}_i$ for all i and Λ_μ to be a diagonal matrix containing the square-roots of the elements of μ:

$$\hat{\mathbf{X}} \triangleq [\hat{\mathbf{X}}_1, \ldots, \hat{\mathbf{X}}_M],$$

$$\Lambda_\mu = \mathbf{diag}([\sqrt{\mu_1}\mathbf{1}_{N_1}^T, \ldots, \sqrt{\mu_M}\mathbf{1}_{N_M}^T]^T).$$

Given these definitions, we can approximate $\mathbf{R} \triangleq \sum_j \mu_j \mathbf{D}_j$ as $\hat{\mathbf{R}}$, which is defined below:

$$\hat{\mathbf{R}} \triangleq \hat{\mathbf{X}}\Lambda_\mu^2\hat{\mathbf{X}}^T, \tag{7.14}$$

Here, $\hat{\mathbf{R}}$ represents the empirical estimate of $\mathbf{R}$. Substituting $\hat{\mathbf{R}}$ for $\mathbf{R}$ in (7.10) gives us the following expression for the QP in Section 7.2.1:

$$\max_{\substack{0\leq\alpha_j\leq C\cdot\hat{p}_j \ \forall j \\ \alpha^T y=0}} \quad 2\sum_j \frac{\alpha_j}{\mu_j} - \alpha^T\Lambda_y\bar{\mathbf{X}}^T(\hat{\mathbf{X}}\Lambda_\mu^2\hat{\mathbf{X}}^T)^{-1}\bar{\mathbf{X}}\Lambda_y\alpha. \tag{7.15}$$

We can now use the *kernel PCA* technique described in *Schoelkopf and Smola* [2002] to obtain a "kernelized" version of the above expression. By the singular value decomposition, we have

$$\hat{\mathbf{X}}\Lambda_\mu^2\hat{\mathbf{X}}^T = \mathbf{U}\Sigma^2\mathbf{U}^T,$$
$$\Lambda_\mu\hat{\mathbf{X}}^T\hat{\mathbf{X}}\Lambda_\mu = \mathbf{V}\Sigma^2\mathbf{V}^T,$$

where $\mathbf{U}$ and $\mathbf{V}$ are the eigenvector matrices for $\hat{\mathbf{X}}\Lambda_\mu^2\hat{\mathbf{X}}^T$ and $\Lambda_\mu\hat{\mathbf{X}}^T\hat{\mathbf{X}}\Lambda_\mu$, respectively, and Σ^2 is the corresponding diagonal matrix of eigenvalues. If $\mathbf{V}\Sigma^2\mathbf{V}^T$ is full-rank, then we can express $\mathbf{U}$ as follows:

$$\mathbf{U} = \hat{\mathbf{X}}\Lambda_\mu\mathbf{V}\Sigma^{-1}. \tag{7.16}$$

The above equation can be used to obtain an expression for the *pseudoinverse* of $\hat{\mathbf{R}}$, which we represent as $\hat{\mathbf{R}}^+$.

$$\begin{aligned}
\hat{\mathbf{R}}^+ &= \mathbf{U}\Sigma^{-2}\mathbf{U}^T, \\
&= \hat{\mathbf{X}}\Lambda_\mu\mathbf{V}\Sigma^{-1}\Sigma^{-2}\Sigma^{-1}\mathbf{V}^T\Lambda_\mu\hat{\mathbf{X}}^T, \\
&= \hat{\mathbf{X}}\Lambda_\mu(\Lambda_\mu\hat{\mathbf{X}}^T\hat{\mathbf{X}}\Lambda_\mu)^{-2}\Lambda_\mu\hat{\mathbf{X}}^T, \\
&= \hat{\mathbf{X}}\Lambda_\mu\Lambda_\mu^{-1}(\hat{\mathbf{X}}^T\hat{\mathbf{X}})^{-1}\Lambda_\mu^{-1}\Lambda_\mu^{-1}(\hat{\mathbf{X}}^T\hat{\mathbf{X}})^{-1}\Lambda_\mu^{-1}\Lambda_\mu\hat{\mathbf{X}}^T, \\
&= \hat{\mathbf{X}}(\hat{\mathbf{X}}^T\hat{\mathbf{X}})^{-1}\Lambda_\mu^{-2}(\hat{\mathbf{X}}^T\hat{\mathbf{X}})^{-1}\hat{\mathbf{X}}^T.
\end{aligned}$$

Substituting $\hat{\mathbf{R}}^+$ for $\mathbf{R}^{-1}$ in (7.10) gives us the following QP:

$$\max_{\substack{0\preceq\alpha\preceq C\cdot\hat{p}_j\ \forall j \\ \alpha^T y=0}} \quad 2\sum_j \frac{\alpha_j}{\mu_j} - \alpha^T\Lambda_y\bar{\mathbf{X}}^T\hat{\mathbf{X}}(\hat{\mathbf{X}}^T\hat{\mathbf{X}}\Lambda_\mu^2\hat{\mathbf{X}}^T\hat{\mathbf{X}})^{-1}\hat{\mathbf{X}}^T\bar{\mathbf{X}}\Lambda_y\alpha. \tag{7.17}$$

As in the conventional SVM, the feature vectors in (7.17) only appear in the form of inner products. Thus, we can apply the *kernel trick* to (7.17), where we replace each inner product $\mathbf{x}_1^T\mathbf{x}_2$ with a kernel function, $k(\mathbf{x}_1, \mathbf{x}_2)$. In general, we can implement the AMC-SVM for any arbitrary feature mapping Φ by plugging in the corresponding kernel function, k. Further details on the kernel trick can be found in Chapter 3.

If we now compare the expression in (7.17) with the expression in (7.15), we see that the two expressions are the same except that each instance of $\bar{\mathbf{X}}$ and $\hat{\mathbf{X}}$ in (7.15) is replaced

with $\hat{\mathbf{X}}^T\bar{\mathbf{X}}$ and $\hat{\mathbf{X}}^T\hat{\mathbf{X}}$ in (7.17). Thus, we can view $\hat{\mathbf{X}}^T\bar{\mathbf{X}}$ in (7.17) as a column matrix of input feature vectors. Under this interpretation, we can use (7.14) to come up with the corresponding equation for $\hat{\mathbf{R}}$. This gives us

$$\hat{\mathbf{R}} \triangleq \hat{\mathbf{X}}^T\hat{\mathbf{X}}\Lambda_\mu^2\hat{\mathbf{X}}^T\hat{\mathbf{X}}.$$

We can now substitute $\hat{\mathbf{R}}$ for $\mathbf{R}$ and $\hat{\mathbf{X}}^T\bar{\mathbf{X}}$ for $\bar{\mathbf{X}}$ in (7.10) to arrive at the expression in (7.17). Here, we note that because $\hat{\mathbf{X}}^T\bar{\mathbf{X}}$ is an $N \times J$ matrix, where N is the total number of training examples and J is the total number of classes, the new "feature vectors" in the columns of $\hat{\mathbf{X}}^T\bar{\mathbf{X}}$ will be N-dimensional. Thus, if the original feature space has a dimensionality of L, where $L > N$, then we can use the empirical approach described in this section to effectively reduce the dimensionality of the input feature vectors from L down to N.

SOCP Formulation

The preceding section describes what we refer to as the "empirical formulation" of the iterated QP, where the covariance matrices are estimated from the training data. We can use a similar set of arguments as those given in Section 7.2.1 to obtain an empirical formulation for the SOCP in 7.2.2. As with the empirical formulation of the iterated QP, the empirical SOCP formulation is "kernelized" in the sense that the feature vectors only appear in the form of inner products. To derive this formulation, we begin by estimating the covariance matrix $\mathbf{C}_j$ as follows:

$$\hat{\mathbf{C}}_j = \frac{1}{\hat{p}_j}\hat{\mathbf{X}}_j\hat{\mathbf{X}}_j^T.$$

Here, $\hat{\mathbf{C}}_j$ represents the empirical estimate of $\mathbf{C}_j$, based on the training data. The matrix $\hat{\mathbf{X}}_j$ is defined in the previous section. Given $\hat{\mathbf{C}}_j$, the corresponding empirical estimate for $\mathbf{D}_j^{\frac{1}{2}}$ is

$$\hat{\mathbf{D}}_j^{\frac{1}{2}} = \hat{\mathbf{X}}_j.$$

Substituting $\hat{\mathbf{D}}_j^{\frac{1}{2}}$ for $\mathbf{D}_j^{\frac{1}{2}}$ in (7.12) yields the following *empirical SOCP* formulation:

$$\begin{aligned}
\min_{\mathbf{v},b,\xi,t} \quad & \sum_j (t_j + C\hat{p}_j\xi_j) && (7.18)\\
\textit{subject to} \quad & \frac{1}{2}(t_j + y_j(\mathbf{v}^T\bar{\mathbf{x}}_j + b) + \xi_j) \geq \|\hat{\mathbf{X}}_j^T\mathbf{v};\ \frac{1}{2}(t_j - y_j(\mathbf{v}^T\bar{\mathbf{x}}_j + b) - \xi_j)\|_2 \quad \forall j,\\
& 1 - \xi_j \leq y_j(\mathbf{v}^T\bar{\mathbf{x}}_j + b) \quad \forall j,\\
& 0 \preceq \xi.
\end{aligned}$$

By using *kernel PCA* as in the previous section, we can obtain the following kernelized formulation of the SOCP:

$$\begin{aligned}
\min_{\mathbf{v},b,\xi,t} \quad & \sum_j (t_j + C\hat{p}_j\xi_j) && (7.19)\\
\textit{subject to} \quad & \frac{1}{2}(t_j + y_j(\mathbf{v}^T\hat{\mathbf{X}}_j^T\bar{\mathbf{x}}_j + b) + \xi_j)\\
& \qquad \geq \|\hat{\mathbf{X}}_j^T\hat{\mathbf{X}}_j\mathbf{v};\ \frac{1}{2}(t_j - y_j(\mathbf{v}^T\hat{\mathbf{X}}_j^T\bar{\mathbf{x}}_j + b) - \xi_j)\|_2 \quad \forall j,\\
& 1 - \xi_j \leq y_j(\mathbf{v}^T\hat{\mathbf{X}}_j^T\bar{\mathbf{x}}_j + b) \quad \forall j,\\
& 0 \preceq \xi.
\end{aligned}$$

Again, as in the conventional SVM, the feature vectors in (7.19) only appear in the form of inner products. Thus, we can apply the *kernel trick* to (7.19), where we replace each inner product $\mathbf{x}_1^T\mathbf{x}_2$ with a kernel function, $k(\mathbf{x}_1, \mathbf{x}_2)$.

7.3 Experiments on Artificial Data

The following section describes a set of experiments where we test the adaptive, multicluster framework introduced in Sections 7.1 and 7.2 on various types of artificial data. In each experiment, we first define one or more distributions on the means and on the covariance matrices for a set Gaussian classes that reside in an N-dimensional feature space. We then draw from these distributions to obtain the set, $S_i = \{(\bar{\mathbf{x}}_{i,1}, \mathbf{C}_{i,1}, y_{i,1}), \ldots, (\bar{\mathbf{x}}_{i,M}, \mathbf{C}_{i,M}, y_{i,M})\}$, where $(\bar{\mathbf{x}}_{i,j}, \mathbf{C}_{i,j}, y_{i,j})$ represents the mean,

covariance matrix, and corresponding label for the jth class in set S_i. For these experiments, we used a one-versus-all setting where each class is assigned a label of -1 except for one randomly chosen "target class," which gets a label of 1. The experiments were performed using the iterated QP formulation.

Given the set S_i, the means and covariance matrices for the Gaussian classes defined by S_i are used to train an AMC-SVM, which we then test on 10,000 random draws taken from the same Gaussian classes. For simplicity, we used the *exact* means and covariance matrices defined in S_i to train each AMC-SVM. The only exception to this is Experiment 3, where the covariance matrix of the target class is assumed to be unknown. We used the following weighted metric to compute classification error:

$$error = \frac{1}{2}\left(\frac{\#\ of\ false\ neg.}{\#\ true\ trials} + \frac{\#\ of\ false\ pos.}{\#\ impostor\ trials}\right).$$

Here, *error* represents the average of the empirical rates of false positives and false negatives. Final results for each experiment were obtained by computing the average of *error* over $\sim$3000 draws of S_i for various values of the SVM hyperparameter, C.

7.3.1 Experiment 1

For our first experiment, we tested the AMC-SVM in a one-versus-all setting with 10 Gaussian classes in a 10-dimensional feature space. In this experiment, the class means and class covariance matrices are both defined to be iid, respectively. The exact distributions of the means and covariance matrices are given below:

$$\bar{\mathbf{x}}_i \sim \mathcal{N}(0,\sigma)^{10} \quad \forall i \in \{1,\ldots,10\},$$
$$\mathbf{C}_i = V_i\, diag(\lambda_i)\, V_i^T \quad \forall i \in \{1,\ldots,10\},$$
$$\lambda_i \sim \mathbf{Unif}(0,1)^{10} \quad \forall i \in \{1,\ldots,10\},$$
$$\sigma = 1.5.$$

Here, the columns of $V_i \in \mathbb{R}^{10\times 10}$ represent a set of orthonormal eigenvectors for the covariance matrix, $\mathbf{C}_i$. The V_i matrices are drawn independently from a uniform distribution

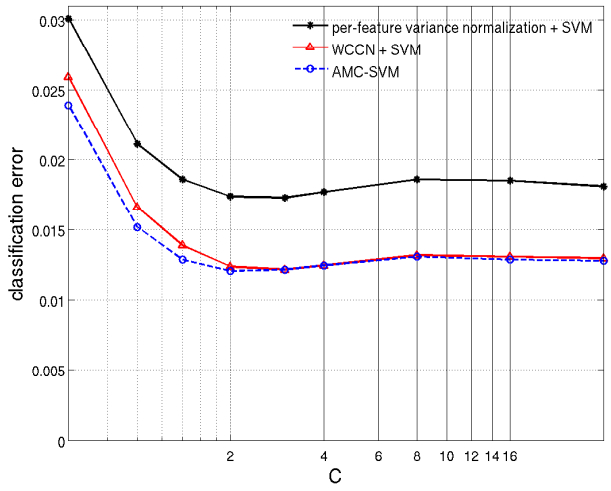

Figure 7.2. Results for Experiment 1.

over all possible sets of orthonormal eigenvectors in $\mathbb{R}^{10}$. Since the V_i matrices serve as eigenvectors for the class covariance matrices, the Gaussian classes in this experiment are completely independent of one another in their spatial orientation (i.e., the directions of their major and minor axes in $\mathbb{R}^{10}$).

Figure 7.2 shows classification results obtained from using the AMC-SVM for various values of the hyperparameter C. The figure also shows two sets of results obtained from a conventional 1-norm, soft margin SVM. These include a set of baseline results where we perform per-feature variance normalization on the input features (i.e., $\mathbf{R} = diag(\mathbf{C})^{-1}$), and another set of results where we perform WCCN (i.e., $\mathbf{R} = \mathbf{C}_W^{-1}$). Note that both the conventional SVM and the AMC-SVM were trained only on the class *means*.

In Figure 7.2, we see that WCCN yields dramatic improvements over the baseline, where the features are scaled to have unit variance. The figure also shows that the AMC-SVM yields modest improvements over WCCN when C is small. However, these improvements become fairly negligible when C is large.

7.3.2 Experiment 2

We performed a second experiment where the Gaussian classes form two major clusters of 5 classes each. In this experiment, the class means within each cluster are correlated with one-another, as are the class covariance matrices. We use the notation, $\bar{\mathbf{x}}_{i,j}$ and $\mathbf{C}_{i,j}$, to represent the mean and covariance matrix for the jth class of cluster i. The exact distributions for the class means and class covariance matrices are defined below:

$$\bar{\mathbf{x}}_{i,j} = 10 \cdot \bar{\mathbf{y}}_{i,0} + \bar{\mathbf{y}}_{i,j} \quad \forall (i,j) \in \{1,2\} \times \{1,\ldots,5\},$$

$$\bar{\mathbf{y}}_{i,j} \sim \mathcal{N}(0,1)^{10} \quad \forall (i,j) \in \{1,2\} \times \{0,\ldots,5\},$$

$$\mathbf{C}_{i,j} = (1-\alpha) \cdot \Sigma_{i,0} + \alpha \cdot \Sigma_{i,j} \quad \forall (i,j) \in \{1,2\} \times \{1,\ldots,5\},$$

$$\Sigma_{i,j} = V_{i,j}\, diag(\lambda_{i,j})\, V_{i,j}^T \quad \forall (i,j) \in \{1,2\} \times \{0,\ldots,5\},$$

$$\lambda_{i,j} \sim \mathbf{Unif}(0,1)^5 \quad \forall (i,j) \in \{1,2\} \times \{0,\ldots,5\},$$

$$\alpha = 0.1.$$

Here, $\bar{\mathbf{y}}_{i,0}$ and $\Sigma_{i,0}$ represent the "primary" mean and covariance matrix for the classes in cluster i. These are combined in a weighted sum with $\bar{\mathbf{y}}_{i,j}$ and $\Sigma_{i,j}$ for all $j \in 1,\ldots,5$. Note that in this experiment, the class means within a given cluster are heavily correlated with one-another, as are the class covariance matrices. The matrix $V_{i,j}$ is distributed in the same way as in Experiment 1.

Figure 7.3 shows classification results for various values of C (see *Hatch* [2006] for a chart of the corresponding numerical results). In this experiment, WCCN and the AMC-SVM both yield large improvements over the baseline results, where we use a conventional SVM after performing per-feature variance normalization. As in Experiment 1, the AMC-SVM significantly outperforms WCCN when the C hyperparameter is small. However, the two systems perform about the same when C is large. For the purpose of training an AMC-SVM, we note from (7.3) that scaling down C is equivalent to scaling up our estimates of the class-conditional covariance matrices—that is, the estimates that we use in training. In other words, reducing C increases the amount of second-order *regularization* that we

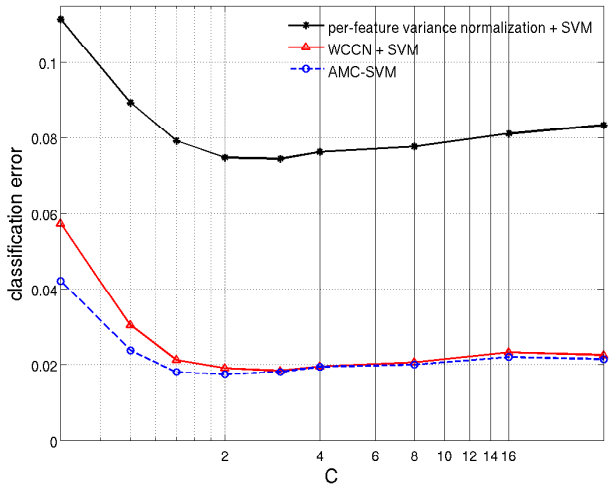

Figure 7.3. Results for Experiment 2.

perform in training. Based on this, and based on the results in Figure 7.2, we might assume that the relative benefits of the AMC-SVM over WCCN are potentially quite significant for cases where a large amount of regularization is required—for example, the case where we only have noisy estimates of the class covariance matrices.

7.3.3 Experiment 3

In many real-world tasks, the amount of training data available for the target class is very limited, whereas the amount of available impostor data is very large. For these tasks, coming up with an empirical estimate of the target class covariance matrix can be difficult, if not impossible. Thus, it may be necessary to estimate the covariance matrix of the target class from the impostor classes—particularly from impostor classes that are highly confusable with the target class. To simulate this scenario, we repeated Experiment 2 for the case where the covariance matrix of target class i is estimated as a weighted average of the covariance matrices of the impostor classes:

$$\mathbf{C}_i = \frac{1}{\sum_{j \in \mathcal{I}} \mu_j \hat{p}_j} \sum_{j \in \mathcal{I}} \mu_j \hat{p}_j \mathbf{C}_j.$$

The averaging scheme given above is based on the definition of $\mathbf{R}$ in equation (7.5). We

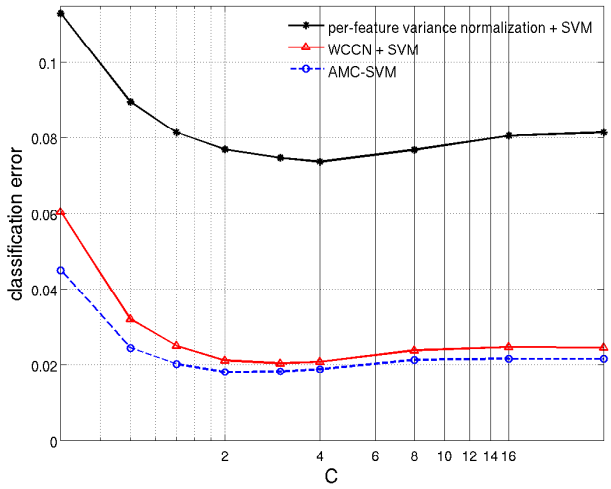

Figure 7.4. Results for Experiment 3.

use μ as the main parameter in defining an estimate of $\mathbf{C}_i$ based on a weighted average of covariance matrices. Insofar as μ_j represents a measure of "confusability" between class j and the target class, this estimate of $\mathbf{C}_i$ assumes that classes that are highly confusable with one-another will tend to have similar covariance matrices. Note that while this assumption may be reasonable for the particular class means and class covariance matrices that we have defined for Experiment 2, we cannot expect this assumption to hold for general datasets.

Results for Experiment 3 are shown in Figure 7.4 along with the corresponding baseline results (a chart of the corresponding numerical results can be found in *Hatch* [2006]). In this experiment, the relative improvement obtained from the AMC-SVM over WCCN is much more significant through the entire range of C values than the improvement obtained in Experiments 1 and in Experiment 2. Thus, we might conclude that the potential benefits of the AMC-SVM approach are particularly significant in cases where information about the target class covariance matrix can be gleaned from impostor classes that lie close to the decision boundary.

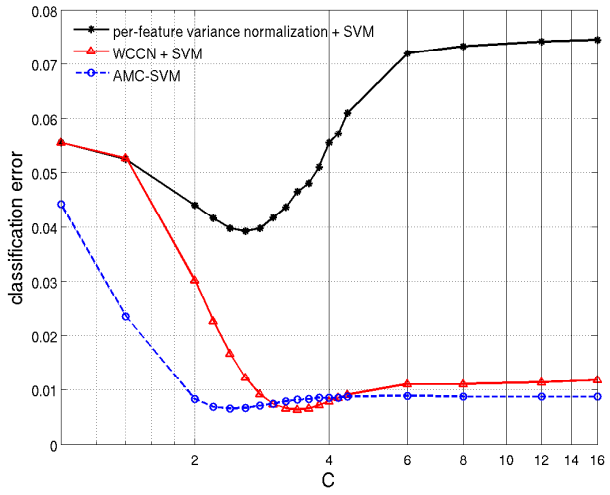

Figure 7.5. Results for Experiment 4.

7.4 Experiment 4

In our 4th and final experiment, we used an artificial Gaussian dataset similar to that of Experiments 2 and 3, except that the dimensionality of the input feature space is increased from 10 to 100. We also changed the distribution of the data so that the classes reside in 5 clusters of 2 classes each. The exact distributions for the class means and class covariance matrices are defined below:

$$\bar{\mathbf{x}}_{i,j} = 10 \cdot \bar{\mathbf{y}}_{i,0} + \bar{\mathbf{y}}_{i,j} \quad \forall (i,j) \in \{1,\ldots,5\} \times \{1,\ldots,2\},$$

$$\bar{\mathbf{y}}_{i,j} \sim \mathcal{N}(0,1)^{100} \quad \forall (i,j) \in \{1,\ldots,5\} \times \{0,\ldots,2\},$$

$$\mathbf{C}_{i,j} = (1-\alpha) \cdot \Sigma_{i,0} + \alpha \cdot \Sigma_{i,j} \quad \forall (i,j) \in \{1,\ldots,5\} \times \{1,\ldots,2\},$$

$$\Sigma_{i,j} = V_{i,j}\, diag(\lambda_{i,j})\, V_{i,j}^T \quad \forall (i,j) \in \{1,\ldots,5\} \times \{0,\ldots,2\},$$

$$\lambda_{i,j} \sim \mathbf{Unif}(0,1)^5 \quad \forall (i,j) \in \{1,\ldots,5\} \times \{0,\ldots,2\},$$

$$\alpha = 0.1.$$

Results for the AMC-SVM are shown in Figure 7.5, along with a set of results obtained from using WCCN and another set of results for the baseline system. Here, we see that the AMC-SVM approach yields large improvements over WCCN for most values of the C

hyperparameter. (However, the WCCN approach in Figure 7.5 outperforms the AMC-SVM when C is approximately between 3 and 4. The minimum classification error achieved by the two techniques is approximately the same.) Based on these results, we might conclude that the AMC-SVM is more robust to sub-optimal values of C than WCCN. Comparing these results with those of Experiment 2, where the input feature space has a dimensionality of 10, we note that the potential benefits of the AMC-SVM over WCCN appear to grow larger as the dimensionality of the input space increases. These results give us reason to believe that the AMC-SVM may yield significant improvements over WCCN on real-world tasks where the dimensionality of the input space is large relative to the number of training examples, and where the optimal value of C is unknown.

7.5 Conclusions

In this chapter, we extend the WCCN approach of Chapter 6 to obtain the so-called *adaptive, multicluster SVM* (AMC-SVM). The AMC-SVM implements an adaptive form of WCCN, where the weights of the class covariance matrices are adapted to the given dataset. This formulation is based on a tighter set of upper bounds on classification error than those used to derive WCCN in Chapters 5 and 6. The AMC-SVM is convex and can be instantiated as either an *iterated QP* or as an *SOCP*. We also show how either instantiation can be "kernelized" in the same way as a conventional SVM. In experiments performed on artificial Gaussian data, the AMC-SVM yields modest, but significant improvements over WCCN when the dimensionality of the input feature space is small compared to the number of classes. These improvements become more substantial as the dimensionality of the feature space is increased.

Chapter 8

Within-Class Covariance Normalization for High-Dimensional Data

In this chapter, we expand on the *within-class covariance normalization* (WCCN) technique that was introduced in Chapter 6. WCCN uses information about class labels from the training data to identify orthonormal directions in feature space that maximize task-relevant information. In this respect, WCCN is similar to other linear transformations such as NAP (*Solomonoff et al.* [2004, 2005]) or linear discriminant analysis (LDA). However, unlike these techniques, WCCN optimally weights each direction in feature space to minimize a particular upper bound on the risk function, $\mathcal{R}(f)$ (*Hatch and Stolcke* [2006]; *Hatch* [2006]). In principle, WCCN can harness whatever task-relevant information is contained in each of the "directions" of the underlying feature space—even directions that are largely dominated by noise.

In the following chapter, we describe a practical procedure for applying WCCN to an SVM-based speaker recognition system where the input feature vectors reside in a high-

dimensional space. Our approach involves using principal component analysis (PCA) to split the original feature space into two subspaces: a low-dimensional, high-energy "PCA space" and a high-dimensional, low-energy "PCA-complement space." After performing WCCN in the PCA space, we concatenate the resulting feature vectors with a weighted version of their corresponding components from the PCA-complement space. Our algorithm provides a practical approach for applying WCCN to large feature sets, where inverting or simply estimating $\mathbf{C}_W$ is impractical for computational reasons. In experiments on SRI International's latest MLLR-SVM speaker verification system (i.e., feature set), our combined WCCN approach achieves relative improvements of up to 22% in equal-error rate (EER) and 28% in minimum DCF below SRI's previous baseline. We also achieve substantial improvements over an MLLR-SVM system that performs WCCN in the PCA space but discards the PCA-complement.

The chapter is organized as follows: In Section 8.1, we summarize the WCCN approach and discuss practical considerations for how to apply WCCN to large feature sets. In Section 8.2, we describe the approach used in *Kajarekar* [2005] for breaking feature vectors down into PCA and PCA-complement components. This is followed by Section 8.3, where we describe the experimental procedure that we use to perform feature normalization and to train SVM-based speaker models. Finally, in Sections 8.4 and 8.6, we describe a set of experiments, provide results, and end with a set of conclusions.

8.1 Within-Class Covariance Normalization

In Chapter 6, we derived WCCN by first constructing a set of upper bounds on the rates of false positives and false negatives in a linear classifier (i.e., a binary classifier that uses a linear or affine decision boundary). Under various conditions, the problem of minimizing these upper bounds with respect to the parameters of the linear classifier leads to a modified formulation of the *hard-margin support vector machine* (SVM) (*Vapnik* [1995]; *Cristianini and Shawe-Taylor* [2000]). Given a generalized linear kernel of the form, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{R} \mathbf{x}_2$,

where $\mathbf{R}$ is a positive semidefinite parameter matrix, this modified SVM formulation implicitly prescribes the parameterization, $\mathbf{R} = \mathbf{C}_W^{-1}$, where $\mathbf{C}_W$ is the expected within-class covariance matrix over all classes. We can represent $\mathbf{C}_W$ mathematically as

$$\mathbf{C}_W \triangleq \sum_{i=1}^{M} p(i) \cdot \mathbf{C}_i,$$

$$\mathbf{C}_i \triangleq \mathbb{E}\,(\mathbf{x}_i - \bar{\mathbf{x}}_i)(\mathbf{x}_i - \bar{\mathbf{x}}_i)^T \qquad \forall i.$$

Here, $\mathbf{x}_i$ represents a random draw from class i, M represents the total number of classes, and $\bar{\mathbf{x}}_i$ represents the expected value of $\mathbf{x}_i$. We use $\mathbf{C}_i$ and $p(i)$ to represent the covariance matrix and the prior probability of class i. (Note that in this chapter, the term, "class" is synonymous with "speaker.") Given $\mathbf{C}_W$, where $\mathbf{C}_W$ is full-rank, we can implement a generalized linear kernel with $\mathbf{R} = \mathbf{C}_W^{-1}$ by using the following feature transformation, Φ:

$$\Phi(\mathbf{x}) \triangleq \mathbf{A}^T \mathbf{x}. \tag{8.1}$$

Here, $\mathbf{A}$ is defined as the Cholesky factorization of $\mathbf{C}_W^{-1}$:

$$\mathbf{A}\mathbf{A}^T \triangleq \mathbf{C}_W^{-1}.$$

We refer to the transformation performed by Φ as *within-class covariance normalization* (WCCN).

In practice, empirical estimates of $\mathbf{C}_W$ are typically quite noisy; thus, a certain amount of smoothing is usually required to make the WCCN approach work. In this chapter, we use the following smoothing model:

$$\hat{\mathbf{C}}_{W,s} \triangleq (1 - \alpha) \cdot \hat{\mathbf{C}}_W + \alpha \cdot \mathbf{I}, \qquad \alpha \in [0, 1]. \tag{8.2}$$

Here, $\hat{\mathbf{C}}_{W,s}$ represents a smoothed version of the empirical expected within-class covariance matrix, $\hat{\mathbf{C}}_W$, and $\mathbf{I}$ represents an $N \times N$ identity matrix where N is the dimensionality of the feature space. The α parameter represents a tunable smoothing weight whose value is between 0 and 1. It is straightforward to show that in the above model, the eigenvectors

of $\hat{\mathbf{C}}_{W,s}$ are constant with respect to α. Thus, we can compute the WCCN feature transformation, Φ, in (8.1) for any value of α without having to recompute the eigenvectors of $\hat{\mathbf{C}}_{W,s}$.

8.1.1 WCCN for High-Dimensional Data

In this chapter, we examine the problem of how to apply WCCN to high-dimensional data sets, where inverting or simply estimating $\hat{\mathbf{C}}_W$ is impractical for computational reasons. For high-dimensional data, we can use kernel principal component analysis (KPCA) to first reduce the dimensionality of the feature space to a more manageable size before performing WCCN. One potential problem with this approach, however, is that by filtering out various orthogonal vectors or "directions" in feature space (i.e., by performing feature reduction), we lose a significant amount of the information contained in the original feature set. To avoid this problem, we use the PCA decomposition described in *Kajarekar* [2005], where the feature space is divided into two subspaces: a space that contains the top N features obtained from performing PCA, and a *PCA-complement* space, which includes all of the information contained in the original features but not in the PCA space. In this chapter, we set N equal to the total number of feature vectors in the training data. Thus, the PCA decomposition retains all of the energy (i.e., variance) of the original training data. Conversely, the PCA-complement space retains *none* of the energy of the training data. For new feature vectors (i.e., test data), the PCA space will tend to have high energy and low dimensionality, while the PCA-complement space will tend to have low energy and high-dimensionality. Since most of the signal energy is confined to the PCA space, our strategy is to perform WCCN on the PCA-space, which has reduced dimensionality, and then concatenate the resulting feature set with the PCA-complement space. This procedure is described in the following sections.

8.2 Kernel PCA and the PCA-Complement

This section provides an overview of kernel PCA and also describes the PCA-complement approach used in *Kajarekar* [2005]. We begin by defining $\mathbf{X}$ to be a column matrix containing scaled, mean-centered versions of the feature vectors in the training set:

$$\mathbf{X} \triangleq \sqrt{\frac{1}{N}} \cdot [(\mathbf{x}_1 - \bar{\mathbf{x}}), \ldots, (\mathbf{x}_N - \bar{\mathbf{x}})].$$

Here $\mathbf{x}_i$ represents the ith training vector, and $\bar{\mathbf{x}}$ represents the average over all N training vectors. Given the above definition, we can represent $\hat{\mathbf{C}}$ (i.e., the empirical covariance matrix of the data) as follows:

$$\begin{aligned} \hat{\mathbf{C}} &= \mathbf{X}\mathbf{X}^T, \\ &\triangleq \mathbf{U}\Sigma^2\mathbf{U}^T. \end{aligned} \tag{8.3}$$

In the second line of the above equation, we define $\mathbf{U}\Sigma^2\mathbf{U}^T$ to be the eigendecomposition of $\hat{\mathbf{C}}$. We can represent the corresponding eigendecomposition for $\mathbf{X}^T\mathbf{X}$ as follows:

$$\mathbf{X}^T\mathbf{X} \triangleq \mathbf{V}\Sigma^2\mathbf{V}^T. \tag{8.4}$$

Here, we define $\mathbf{V}$ to be a column matrix containing the eigenvectors of $\mathbf{X}^T\mathbf{X}$ and Σ^2 to be a diagonal matrix containing the corresponding eigenvalues. If $\mathbf{X}^T\mathbf{X}$ is full-rank, then we can combine (8.3) with (8.4) to arrive at the following expression for $\mathbf{U}$, the eigenvector matrix of $\hat{\mathbf{C}}$:

$$\mathbf{U} = \mathbf{X}\mathbf{V}\Sigma^{-1}. \tag{8.5}$$

The columns of $\mathbf{U}$ represent the set of all eigenvectors of $\hat{\mathbf{C}}$ whose corresponding eigenvalue is non-zero. Thus, we can perform PCA by projecting the input feature vectors onto the column vectors of $\mathbf{U}$. This leads to the following feature transformation, Φ_{PCA}:

$$\begin{aligned} \Phi_{PCA}(\mathbf{x}) &\triangleq \mathbf{U}^T\mathbf{x}, \\ &= \Sigma^{-1}\mathbf{V}^T\mathbf{X}^T\mathbf{x}. \end{aligned} \tag{8.6}$$

This transformation reduces the dimensionality of the underlying feature space down to N features, where N is the size of the training set. Since the input feature vectors appear in the form of inner products, which can be replaced with kernel functions, this feature transformation is referred to as *kernel PCA* (*Shawe-Taylor and Cristianini* [2004]).

We use $\Phi_{\overline{PCA}}$ to represent the feature transformation for the PCA-complement space, which is defined as follows:

$$\Phi_{\overline{PCA}}(\mathbf{x}) \triangleq (\mathbf{I} - \mathbf{U}\mathbf{U}^T)\mathbf{x}. \tag{8.7}$$

The PCA-complement space represents the portion of the original feature space that is orthogonal to the training set. Thus, $\Phi_{\overline{PCA}}(\mathbf{x}) = \mathbf{0}$ (i.e., a null vector) for all $\mathbf{x}$ in the training set.

8.3 Experimental Procedure

The experiments in this chapter compare two different feature normalizations: WCCN and standard *covariance normalization* (CN), where $\mathbf{R} = \hat{\mathbf{C}}_s^{-1}$. (Here, $\hat{\mathbf{C}}_s$ represents a smoothed version of $\hat{\mathbf{C}}$, the empirical covariance matrix of the training data.) Since $\Phi_{\overline{PCA}}(\mathbf{x}) = \mathbf{0}$ for all $\mathbf{x}$ in the training set, we have no way of coming up with a meaningful estimate of the covariance matrix for the PCA-complement (any empirical covariance estimate will simply be $\mathbf{0}$). Thus, WCCN and standard CN are only applied to the PCA feature set. The normalized PCA features are then concatenated with a weighted version of the PCA-complement to form the final feature representation.

Our experimental procedure is summarized below:

1. Perform per-feature within-class *variance* normalization on all of the input features (i.e., scale all features to have an average within-class variance of one on the training data). The resulting features provide us with a first-cut approximation of what we would obtain by performing full WCCN on the original feature set. This is simply a preprocessing step for performing KPCA, which is not invariant to scaling operations

on the input features. Note that the smoothing model of (8.2) is also not invariant to scaling operations.

2. Compute $\Phi_{PCA}(\mathbf{x})$ for every feature vector $\mathbf{x}$ in the training and test sets. This gives us the PCA feature set.

3. Compute $\Phi_{\overline{PCA}}(\mathbf{x})$ for every feature vector $\mathbf{x}$ in the training and test sets. This gives us the PCA-complement feature set.

4. Perform either within-class covariance normalization (WCCN) or standard covariance normalization (CN) on the PCA feature set. Both normalizations can be represented in the form of a matrix multiplication. We use the smoothing model shown in equation (8.2) for both WCCN and standard CN. The smoothing parameter α is tuned on a set of held-out cross-validation data.

5. Concatenate a scaled version of the normalized PCA feature set with a scaled version of the PCA-complement feature set to arrive at our final feature representation, Φ:

$$\Phi(\mathbf{x}) \triangleq \begin{bmatrix} (1-\beta) \cdot \mathbf{A}^T \Phi_{PCA}(\mathbf{x}) \\ \beta \cdot \Phi_{\overline{PCA}}(\mathbf{x}) \end{bmatrix}, \qquad \beta \in [0,1]. \tag{8.8}$$

   Here, $\mathbf{A}^T$ represents the transformation matrix derived in step 4 to perform either WCCN or standard CN on the PCA feature set. Thus, $\mathbf{A}^T \Phi_{PCA}(\mathbf{x})$ represents the normalized PCA component of feature vector $\mathbf{x}$. We use the parameter β to control the relative weight applied to the two feature sets (i.e., the PCA set and the PCA-complement set). This parameter is tuned on a held-out cross-validation set.

6. Use the final feature representation to train and test SVM-based speaker models.

A diagram of this procedure is shown in Figure 8.1. Given a standard linear kernel, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{x}_2$, it's fairly straightforward to show that when $\beta = 0.5$ and $\mathbf{A} = \mathbf{I}$ (i.e., $\mathbf{A}$ is the idenitity matrix), then the following equality holds for any pair of input feature vectors, $\mathbf{x}_1$ and $\mathbf{x}_2$:

$$k(\mathbf{x}_1, \mathbf{x}_2) = 4 \cdot k(\Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2)). \tag{8.9}$$

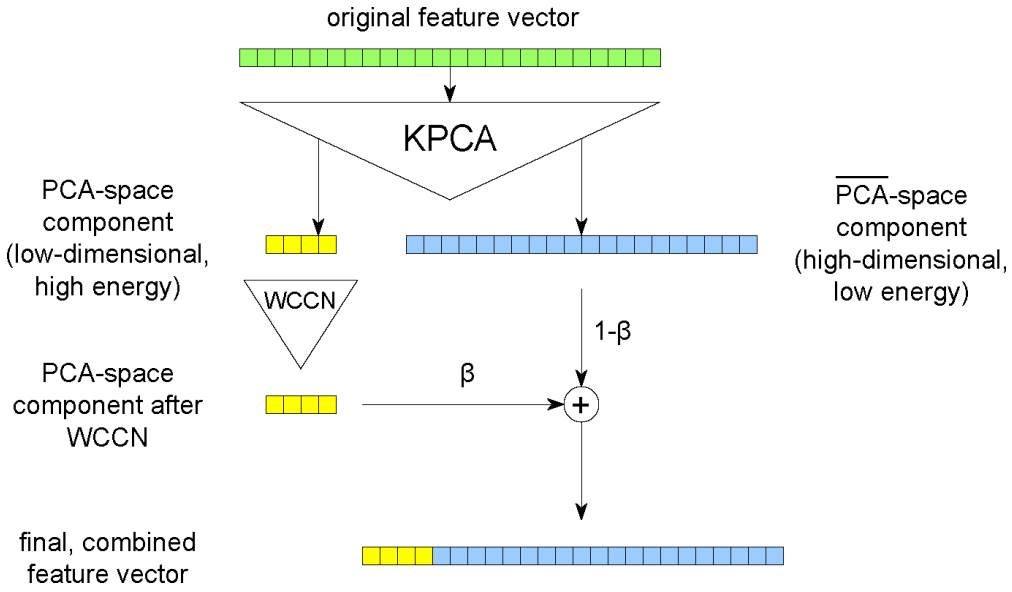

Figure 8.1. Diagram of WCCN procedure for high-dimensional data. The "+" sign in the above figure represents a concatenation operator.

The equality in (8.9) follows directly from the definitions for Φ, Φ_{PCA}, and $\Phi_{\overline{PCA}}$ in equations (8.8), (8.6), and (8.7). Equation (8.9) shows that when $\beta = 0.5$ and $\mathbf{A} = \mathbf{I}$, then applying the feature transformation, Φ, to the input feature vectors does not affect the kernel function k beyond a scaling factor. Thus, by concatenating the PCA set with the PCA-complement set, we preserve all of the information contained in the original feature set, at least for the purpose of computing linear kernels.

8.4 Experiments and Results

In this section, we describe the tasks, datasets, and features used in our experiments. The results of these experiments are discussed in Section 8.4.4.

8.4.1 MLLR-SVM System

We used an MLLR-SVM system similar to the one described in *Stolcke et al.* [2006] to compute feature vectors for our experiments. The MLLR-SVM system uses speaker

adaptation transforms from SRI's DECIPHER speech recognition system as features for speaker verification. A total of 8 affine transforms are used to map the Gaussian mean vectors from speaker-independent to speaker-dependent speech models. The transforms are estimated using maximum-likelihood linear regression (MLLR), and can be viewed as a text-independent encapsulation of the speaker's acoustic properties. For every conversation side, we compute a total of 24960 transform coefficients, which are used as features. Note that this system uses twice as many features as the original MLLR-SVM system described in *Stolcke et al.* [2005]; *Hatch and Stolcke* [2006]. The input feature vectors are identical to those used in *Stolcke et al.* [2006]. However, besides applying the feature transformation Φ to the input feature vectors, our system differs from the MLLR-SVM system used in *Stolcke et al.* [2006] in the following ways: 1) our system does not apply rank normalization (*Stolcke et al.* [2005]) to the input feature vectors and 2) our system does not apply TNORM (*Auckenthaler et al.* [2000]) to the output SVM scores. We have yet to experiment with applying these normalizations to a system that uses WCCN.

8.4.2 Task and Data

Experiments were performed on the 1-conversation training condition of two NIST-defined tasks: SRE-2004 and a subset of SRE-2003. Note that these tasks and datasets are the same as those described in previous reports (*Stolcke et al.* [2006]; *Hatch and Stolcke* [2006]). The SRE-2003 subset was divided into two splits of disjoint speaker sets, both comprised of ∼3600 conversation sides and ∼300 speakers. Each split comprises ∼580 speaker models and ∼9800 speaker trials. These splits were alternately used for training (i.e., computing covariance estimates and feature transformations) and for testing. We used SRE-2004 to tune α and β for testing on SRE-2003, and vice-versa. To simplify the tuning process, α was optimized for the case where $\beta = 0$. The resulting α parameter was then held fixed while tuning β. Further details on the tasks and datasets can be found in *Stolcke et al.* [2006].

8.4.3 SVM Training

We used SVM^{light} (*Joachims* [1999]) to train SVM-based speaker models for each task. Each speaker model was trained with a linear kernel using the default value of the SVM hyperparameter C. A held-out dataset composed of 425 conversation sides taken from the Switchboard-2 corpus and 1128 conversation sides taken from the Fisher corpus was used as negative examples for the SVM training.

8.4.4 Results

Table 8.1 shows results on the MLLR-SVM system for various feature representations. Here, the labels "WCCN" and "CN" denote within-class covariance normalization and standard covariance normalization, where α is tuned on the cross-validation set. The β parameter is optimized on the cross-validation set for systems that are labeled "$\overline{\text{PCA}}$." For systems that are *not* labeled "$\overline{\text{PCA}}$," β is set equal to zero (i.e., the PCA-complement is omitted from the final feature representation). The "baseline" label represents the MLLR-SVM system without any feature normalization.

As shown in Table 8.1, the WCCN approach provides improvements that are quite substantial, at least in most cases, over standard CN (see the "improvement over PCA+CN+$\overline{\text{PCA}}$" results). It's worth noting that the improvements obtained over the baseline are significantly larger on SRE-2003 than on SRE-2004. However, this is to be expected, since the feature transformations and normalizations used in these experiments were trained only on held-out SRE-2003 data, which represents a different set of channel and recording conditions than SRE-2004.

We note that the "PCA," "PCA+CN," and "PCA+WCCN" results are all obtained from PCA feature sets whose dimensionality is reduced to ∼3600 (i.e., the number of training examples in each split of the SRE-2003 subset). In spite of this reduced dimensionality, the

Φ	SRE-03 subset		SRE-04	
	EER%	DCF	EER%	DCF
baseline	2.91	0.117	5.97	0.282
PCA	3.89	0.158	7.35	0.318
PCA+CN	2.92	0.123	6.43	0.289
PCA+WCCN	2.30	0.108	5.52	0.260
PCA+$\overline{\text{PCA}}$	2.91	0.117	5.97	0.282
PCA+CN+$\overline{\text{PCA}}$	2.33	0.092	5.87	0.266
PCA+WCCN +$\overline{\text{PCA}}$	2.08	0.091	5.27	0.247
improvement over baseline	**28.5**%	**22.2**%	**11.7**%	**12.4**%
improvement over PCA+WCCN	**9.6**%	**15.7**%	**4.5**%	**5.0**%
improvement over PCA+CN+$\overline{\text{PCA}}$	**10.7**%	**1.1**%	**10.2**%	**7.1**%

Table 8.1. EERs and minimum DCFs for various feature transformations/normalizations on the MLLR-SVM system. Here, "baseline" represents the raw MLLR-SVM system without any feature normalization. The labels "WCCN" and "CN" denote within-class covariance normalization and standard covariance normalization, and "$\overline{\text{PCA}}$" denotes a system that uses the PCA-complement with β optimized on the cross-validation set. The "improvement" entries represent the relative improvement of PCA+WCCN+$\overline{\text{PCA}}$ over the given system.

"PCA+WCCN" system significantly outperforms the "baseline" system, where each feature vector is composed of 24960 features.

Table 8.1 also shows that adding the PCA-complement to the PCA feature set leads to significant relative reductions in error rate (see the "improvement over PCA+WCCN" results). To the best of our knowledge, the results for the "PCA+WCCN+$\overline{\text{PCA}}$" system are the best recorded so far in the literature for an MLLR-SVM system. Even without using rank normalization or TNORM—two techniques used in *Stolcke et al.* [2006] which should presumably lead to reductions in error rate (we have not yet integrated these normalizations into our system)—our system outperforms the MLLR-SVM system in *Stolcke et al.* [2006] by at least 15% on the SRE-2003 subset and by a smaller, but still significant margin on SRE-2004. These experiments point to the utility of using WCCN in conjunction with the PCA-complement when training SVM-based speaker models.

8.5 Experiment 2: Comparison between WCCN and NAP

In this section, we report some recent results from experiments that directly compare WCCN with the NAP approach described in *Solomonoff et al.* [2004, 2005] and in Section 4.6.3. These experiments were performed by Sachin Kajarekar and Andreas Stolcke of SRI International and are reported in *Kajarekar and Stolcke* [2007]. The experiments compare the performance of the WCCN approach described in Chapter 6 versus NAP on SRI International's most recent MLLR-SVM system. Each experiment involves two trainingsets: one to estimate covariance matrices and another to serve as background data when training SVM-based speaker models. The speaker models are trained and tested on the 1-conversation-side condition of both the SRE2005 and SRE2006 speaker verification tasks. The α and β parameters for WCCN and the N parameter for NAP (i.e., the dimensionality of the output feature space after performing NAP) are optimized on SRE2005 and then tested on both SRE2005 and on SRE2006. Thus, the results for SRE2005 technically involve some amount of "cheating," since the testset is used to tune parameters. The experiments in *Hatch and Stolcke* [2006] and in Chapter 6 differ from those described in Section 8.4.4 in the following ways:

1. The training set that is used to compute covariance matrices is also used to perform rank-normalization on the input feature vectors.

2. The experiments use different training sets than those used in Section 8.4.4.

A set of results for WCCN and for NAP are provided in Table 8.2 and Table 8.3.

The best results for each testset are listed in bold type. The figures in parenthesis represent cheating results for SRE2006, where the α, β, and N parameters are tuned on the testset. The parameters for the non-parenthesized results are tuned on SRE2005. The following table lists the relative improvement obtained by using WCCN over NAP. Table 8.4 shows that WCCN achieves modest but significant improvements on all test conditions except for one: the results degrade fairly significantly for the case where the covariance

BKG data	Intersession variability estimated on	SRE05 (English) (DEV set)		SRE06 (English) (EVAL set)	
		%EER	min DCF	%EER	min DCF
Fisher	baseline	5.872	0.190	4.639	0.224
	SRE03	5.066	0.154	4.314	0.198
	SRE04	**5.056**	**0.147**	4.477	0.216
SRE04	baseline	6.189	0.200	4.315	0.197
	SRE03	5.219	0.162	3.776	0.173
	SRE04	5.103	0.157	**3.603** (**3.452**)	**0.166** (**0.162**)

Table 8.2. EERs and minimum DCFs obtained from applying WCCN to an MLLR-SVM system.

BKG data	Intersession variability estimated on	SRE05 (English) (DEV set)		SRE06 (English) (EVAL set)	
		%EER	min DCF	%EER	min DCF
Fisher	baseline	5.872	0.190	4.639	0.224
	SRE03	5.653	0.166	4.423	0.206
	SRE04	**5.470**	**0.158**	3.999	0.196
SRE04	baseline	6.189	0.200	4.315	0.197
	SRE03	5.744	0.172	3.831	0.180
	SRE04	5.664	0.163	**3.614** (**3.567**)	**0.170** (**0.167**)

Table 8.3. EERs and minimum DCFs obtained from applying NAP to an MLLR-SVM system.

matrices and rank-normalization are estimated on SRE2004, the background data is drawn from the Fisher dataset, and testing is performed on SRE2006. The latter degradation is somewhat surprising, especially given that the same combination of trainingsets yields an improvement for WCCN on SRE2005.

8.6 Conclusions

In this chapter, we have described a practical procedure for applying within-class covariance normalization (WCCN) to an MLLR-SVM speaker verification system where the feature vectors reside in a high-dimensional space. When applied to a state-of-the-art MLLR-SVM speaker verification system, this approach achieves improvements of up to

BKG data	Intersession variability estimated on	SRE05 (English) (DEV set)		SRE06 (English) (EVAL set)	
		EER	min DCF	EER	min DCF
Fisher	baseline	—	—	—	—
	SRE03	10.38%	7.23%	2.46%	3.88%
	SRE04	**7.57**%	**6.96**%	-11.95%	-10.20%
SRE04	baseline	—	—	—	—
	SRE03	9.14%	5.81%	1.44%	3.89%
	SRE04	9.90%	3.68%	**0.30**% (**3.22**%)	**2.35**% (**2.99**%)

Table 8.4. Relative improvements in EER and minimum DCF obtained from using WCCN over NAP.

22% in EER and 28% in minimum decision cost function (DCF) over our previous baseline. We also achieve substantial improvements over an MLLR-SVM system that performs WCCN on the PCA set but discards the PCA-complement. These results point to the utility of using WCCN in conjunction with the PCA-complement when training SVM-based speaker models.

This chapter also provides results for experiments that compare WCCN against the *nuisance attribute projection* (NAP) approach described in *Solomonoff et al.* [2004, 2005] and in Section 4.6.3. In these experiments, WCCN typically outperforms NAP by a modest but significant margin when applied to an MLLR-SVM speaker verification system.

Chapter 9

Summary and Conclusions

In this dissertation, we examine the problem of kernel optimization for binary classification tasks where the training data are partitioned into multiple, disjoint classes. The dissertation focuses specifically on the field of speaker verification, which can be framed as a one-versus-all (OVA) decision task involving a target speaker and a set of impostor speakers.

The main result of this dissertation is a new framework for optimizing *generalized linear kernels* of the form, $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{R} \mathbf{x}_2$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are input feature vectors, and $\mathbf{R}$ is a positive semidefinite parameter matrix. Our framework is based on using first and second-order statistics from each class (i.e., speaker) in the data to construct an upper bound on classification error in a linear classifier. Minimizing this bound leads directly to a new, modified formulation of the 1-norm, soft-margin support vector machine (SVM). We refer to this new, modified SVM formulation as the *adaptive, multicluster SVM* (AMC-SVM). The AMC-SVM differs from the conventional soft-margin SVM in *Vapnik* [1995] in the following ways:

1. The AMC-SVM implicitly prescribes a solution for the $\mathbf{R}$ parameter matrix in a generalized linear kernel.

2. The AMC-SVM follows directly from minimizing a particular upper bound on classification error. On the other hand, Vapnik's soft-margin SVM formulation is based on appending *slack variables* to the hard-margin SVM.

3. The C hyperparameter is exactly specified in the AMC-SVM but is undetermined in the conventional soft-margin SVM.

Unlike most other kernel learning techniques in the literature, the AMC-SVM uses information about clusters that reside within the given target and impostor data to obtain tighter bounds on classification error than those obtained in conventional SVM-based approaches. This use of cluster information makes the AMC-SVM particularly well-suited to tasks that involve binary classification of multiclass data—for example, the speaker verification task—where each class (i.e., speaker) can be treated as a separate cluster.

In OVA training settings, we show that the AMC-SVM can, under certain conditions, be formulated to yield a single, fixed kernel function that applies universally to any choice of target speaker. Since this kernel function is linear, we can implement it by applying a single linear feature transformation to the input feature space. This feature transformation performs what we refer to as within-class covariance normalization (WCCN) on the input feature vectors. The dissertation describes a set of experiments where WCCN yields large reductions in classification error over other normalization techniques on a state-of-the-art SVM-based speaker verification system.

Bibliography

Andrews, W. D., M. A. Kohler, and J. P. Campbell, Phonetic Speaker Recognition, in *proceedings of Eurospeech*, pp. 149–153, 2001.

Andrews, W. D., M. A. Kohler, J. P. Campbell, J. J. Godfrey, and J. Hernandez-Cordero, Gender-dependent phonetic refraction for speaker recognition, in *proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. I, pp. 149–153, 2002.

Auckenthaler, R., M. Carey, and H. Lloyd-Thomas, Score normalization for text-independent speaker verification systems, in *Digital Signal Processing*, vol. 10, San Juan, Puerto Rico, 2000.

Bach, F. R., G. R. G. Lanckriet, and M. I. Jordan, Multiple kernel learning, conic duality, and the SMO algorithm, in *proceedings of the International Conference on Machine Learning (ICML)*, 2004.

Boakye, K., and B. Peskin, Text-Constrained Speaker Recognition on a Text-Independent Task, in *proceedings of the IEEE Odyssey 2004 Speaker and Language Recognition Workshop*, Toledo, Spain, 2004.

Boser, B. E., I. M. Guyon, and V. N. Vapnik, A Training Algorithm for Optimal Margin Classifiers, in *proceeding of the Fifth Annual ACM Workshop on COLT*, pp. 144–152, ACM Press, Pittsburgh, PA, 1992.

Boyd, S., and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, 2004.

Bradley, P. S., and O. L. Mangasarian, Feature Selection via Concave Minimization and Support Vector Machines, in *proceedings of the International Conference on Machine Learning (ICML)*, vol. 15, Morgan Kaufmann, San Francisco, CA, 1998.

Campbell, W. M., A Sequence Kernel and its Application to Speaker Recognition, in *Advances in Neural Information Processing Systems (NIPS) 14*, 2001.

Campbell, W. M., Generalized Linear Discriminant Sequence Kernels for Speaker Recognition, in *proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002.

Campbell, W. M., J. P. Campbell, D. A. Reynolds, D. A. Jones, and T. R. Leek, Phonetic speaker recognition with support vector machines, in *Advances in Neural Information Processing Systems (NIPS) 16*, 2003.

Chapelle, O., V. Vapnik, O. Bousquet, and S. Mukherjee, Choosing multiple parameters for support vector machines, in *Machine Learning 46*, 2002.

Cristianini, N., and J. Shawe-Taylor, *Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, Cambridge, 2000.

Davis, S., and P. Mermelstein, Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *28*, 357–366, 1980.

Doddington, G., Speaker recognition based on idiolectal differences between speakers, in *proceedings of Eurospeech*, pp. 2521–2524, 2001.

Ferrer, L., E. Shriberg, S. S. Kajarekar, A. Stolcke, K. Sonmez, A. Venkataraman, and H. Bratt, The Contribution of Cepstral and Stylistic Features to SRI's 2005 NIST Speaker Recognition Evaluation System, in *proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Toulouse, France, 2005a.

Ferrer, L., K. Sonmez, and S. Kajarekar, Class-dependent Score Combination for Speaker Recognition, in *proceedings of the Eurospeech*, Lisbon, 2005b.

Fukunaga, K., *Introduction to Statistical Pattern Recognition*, Academic Press, New York, 1990.

Garcia-Romero, D., J. Fierrez-Aguilar, J. Ortega-Garcia, and J. Gonzalez-Rodriguez, Support Vector Machine Fusion of Idiolectal and Acoustic Speaker Information in Spanish Conversational Speech, in *proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Hong Kong, 2003.

Gauvain, J. L., and C. H. Lee, Maximum A Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains, *IEEE Transactions on Speech and Audio Processing*, *2*, 291–298, 1994.

Grandvalet, Y., and S. Canu, Adaptive scaling for feature selection in SVMs, in *Advances in Neural Information Processing Systems (NIPS)*, 2003.

Hastie, T., R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2001.

Hatch, A., Adaptive linear kernels for binary classification of multicluster data, in *Technical Report*, 2006.

Hatch, A., and A. Stolcke, Generalized linear kernels for one-versus-all classification: application to speaker recognition, in *proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Toulouse, France, 2006.

Hatch, A., B. Peskin, and A. Stolcke, Improved Phonetic Speaker Recognition Using Lattice Decoding, in *proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2005.

Hatch, A., S. Kajarekar, and A. Stolcke, Within-Class Covaraince Normalization for SVM-based Speaker Verification, in *proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Pittsburgh, PA, 2006.

Jebara, T., and T. Jaakkola, Feature Selection and Dualities in Maximum Entropy Discrimination, in *Uncertainty in Artificial Intelligence*, 2000.

Jin, Q., J. Navratil, D. Reynolds, J. Campbell, W. Andrews, and J. Abramson, Combining cross-stream and time dimensions in phonetic speaker recognition, in *proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003.

Joachims, T., Making large-scale SVM learning practical, in *Advances in kernel methods — support vector learning*, edited by B. Schoelkopf, C. Burges, and A. Smola, MIT-press, 1999.

Kajarekar, S., Four weightings and a fusion: a cepstral-svm system for speake r recognition, in *proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, San Juan, Puerto Rico, 2005.

Kajarekar, S., and A. Stolcke, NAP and WCCN: Comparison of Approaches Using MLLR-SVM Speaker Verification System, in *to appear in proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Honolulu, Hawaii, 2007.

Kajarekar, S., L. Ferrer, A. Venkataraman, K. Sonmez, E. Shriberg, A. Stolcke, and R. R. Gadde, Speaker recognition using prosodic and lexical features, in *proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 19–24, 2003.

Kajarekar, S., L. Ferrer, E. Shriberg, K. Sonmez, A. Stolcke, and A. Venkataraman, SRI's NIST 2005 Speaker Recognition Evaluation System, in *presentation at the NIST Speaker Recognition Evaluation Workshop*, Montreal, 2005a.

Kajarekar, S., L. Ferrer, E. Shriberg, K. Sonmez, A. Stolcke, A. Venkataraman, and J. Zheng, SRI's 2004 NIST Speaker Recognition Evaluation System, in *proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2005b.

Klusacek, D., J. Navratil, D. A. Reynolds, and J. P. Campbell, Conditional pronunciation modeling in speaker detection, in *proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. IV, pp. 804–807, 2003.

Lanckriet, G. R. G., L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan, A Robust Minimax Approach to Classification, *Journal of Machine Learning Research (JMLR)*, *3*, 555–582, 2002.

Lanckriet, G. R. G., N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, Learning the kernel matrix with semidefinite programming, *Journal of Machine Learning Research (JMLR)*, *5*, 27–72, 2004.

Leggetter, C., and P. Woodland, Maximum Likelihood Linear Regression for Speaker Adaptation of HMMs, *Computer Speech and Language*, *9*, 171–186, 1995.

Marshall, A. W., and I. Olkin, Multivariate Chebyshev Inequalities, *Annals of Mathematical Statistics*, *31*, 1001–1014, 1960.

Mirghafori, N., A. O. Hatch, S. Stafford, K. Boakye, D. Gillick, and B. Peskin, ICSI's 2005 Speaker Recognition System, in *proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2005.

Navratil, J., Q. Jin, W. Andrews, and J. Campbell, Phonetic speaker recognition using maximum likelihood binary decision tree models, in *proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003.

Neal, R. M., Bayesian Learning for Neural Networks, in *Lecture Notes in Statistics*, Springer, 1996.

NIST, *The NIST Year 2005 Speaker Recognition Evaluation Plan*, NIST, 2005.

Ong, C. S., A. Smola, and R. Williamson, Hyperkernels, in *Advances in Neural Information Processing Systems (NIPS)*, 2003.

Rabiner, L., and B. H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

Reynolds, D., Comparison of Background Normalization Methods for Text-Independent Speaker Verification, in *proceedings of Eurospeech*, pp. 963–966, Rhodes, Greece, 1997.

Reynolds, D., T. Quatieri, and R. Dunn, Speaker Verification Using Adapted Mixture Models, in *Digital Signal Processing*, vol. 10, pp. 181–202, 2000.

Reynolds, D., et al., The SuperSID project: exploting high-level information for high-accuracy speaker recognition, in *proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003.

Rosenberg, A. E., C. H. Lee, and F. K. Soong, Sub-Word Unit Talker Verification Using Hidden Markov Models, in *proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 269–272, 1990.

Rosenberg, A. E., C. H. Lee, and S. Gokeen, Connected Word Talker Recognition Using Whole Word Hidden Markov Models, in *proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 381–384, 1991.

Schoelkopf, B., and A. Smola, *Learning with Kernels*, MIT Press, Cambridge, Massachusetts, 2002.

Shawe-Taylor, J., and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, Cambridge, 2004.

Shriberg, E., L. Ferrer, A. Venkataraman, and S. Kajarekar, SVM Modeling of "SNERF-grams" for Speaker Recognition, in *proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 2004.

Solomonoff, A., C. Quillen, and W. Campbell, Channel Compensation For SVM Speaker Recognition, in *proceedings of Odyssey: The Speaker and Language Recognition Workshop*, Toledo, Spain, 2004.

Solomonoff, A., W. Campbell, and I. Boardman, Advances In Channel Compensation For SVM Speaker Recognition, in *proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Philadelphia, PA, 2005.

Sonnenburg, S., G. Raetsch, and C. Schaefer, A general and efficient multiple kernel learning algorithm, in *Advances in Neural Information Processing Systems (NIPS)*, 2005.

Stolcke, A., L. Ferrer, S. Kajarekar, E. Shriberg, and A. Venkataraman, MLLR transforms as features in speaker recognition, in *proceedings of Interspeech*, 2005.

Stolcke, A., L. Ferrer, and S. Kajarekar, Improvements in MLLR-Transform-based Speaker Recognition, in *proceedings of the IEEE Odyssey 2006 Speaker and Language Recognition Workshop*, San Juan, Puerto Rico, 2006.

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

Weston, J., S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, Feature Selection for SVMs, in *Advances in Neural Information Processing Systems (NIPS)*, vol. 13, MIT Press, 2000.

Appendix A

Derivation of Bounds

The following derivation proves the upper bounds on $\mathcal{R}(f)$ in (6.3) and in (7.3). These upper bounds appear in Theorem 11 and in Theorem 13, respectively.

Proof. We begin with the inequality given in (7.1):

$$\mathbf{1}(y_j f(\mathbf{x}_j) < 0) \leq B(\mathbf{x}_j\,;\,\Theta_j) \quad : \quad 0 < \mu_j \leq 1. \tag{A.1}$$

Here, Θ_j represents a set of parameters for class j, and $B(\mathbf{x}_j\,;\,\Theta_j)$ represents a bounding function. These are defined as follows:

$$\Theta_j \triangleq \{\mathbf{v}, b, \mu_j, y_j, \bar{\mathbf{x}}_j\},$$

$$B(\mathbf{x}_j\,;\,\Theta_j) \triangleq \left(\frac{f(\mathbf{x}_j) - f(\bar{\mathbf{x}}_j)}{\max\left\{\frac{1}{\mu_j}, y_j f(\bar{\mathbf{x}}_j)\right\}}\right)^2 \cdot \mathbf{1}(y_j f(\mathbf{x}_j) \leq y_j f(\bar{\mathbf{x}}_j)) + 2 \cdot (1 - \mu_j y_j f(\bar{\mathbf{x}}_j))_+.$$

In the above equation, we use the shorthand, $(a)_+$, to represent $\mathbf{1}(a > 0) \cdot a$. An illustration of the bound in (A.1) is provided in Figure 7.1. Note that the bound is only valid if $\mu \succ 0$. For the following derivation, we will constrain μ as $0 \prec \mu \preceq 1$. Taking the expectation of both sides of the inequality in (A.1) over all j gives us the following bound on $\mathcal{R}(f)$. Note that in the following derivation, we assume that $\mathbf{x}_j$ is symmetrically distributed about its mean (*i.e.*, $p(\mathbf{x}_j - \bar{\mathbf{x}}_j = \delta) = p(\mathbf{x}_j - \bar{\mathbf{x}}_j = -\delta)$ for all $(j, \delta) \in \{1, \ldots, J\} \times \mathbb{R}^L$, where L is

the dimensionality of the feature space, and J is the total number of classes.

$$\begin{aligned}
\mathcal{R}(f) &\leq \mathop{\mathbb{E}}_{j} B(\mathbf{x}_j\,;\,\Theta_j) \quad : \quad 0 < \mu_j \leq 1 \quad \forall j \\
&= \mathop{\mathbb{E}}_{j} \left(\frac{f(\mathbf{x}_j) - f(\bar{\mathbf{x}}_j)}{\max\left\{\frac{1}{\mu_j}, y_j f(\bar{\mathbf{x}}_j)\right\}} \right)^2 \cdot \mathop{\mathbb{E}}_{j} \mathbf{1}(y_j f(\mathbf{x}_j) \leq y_j f(\bar{\mathbf{x}}_j)) + 2 \cdot \mathop{\mathbb{E}}_{j} (1 - \mu_j y_j f(\bar{\mathbf{x}}_j))_+ \\
&\quad \textit{subject to} \qquad 0 < \mu_j \leq 1 \quad \forall j. \qquad \text{(A.2)} \\
&= \frac{1}{2} \cdot \sum_j \hat{p}_j \frac{\mathbb{E}\,(f(\mathbf{x}_j) - f(\bar{\mathbf{x}}_j))^2}{(\max\left\{\frac{1}{\mu_j}, y_j f(\bar{\mathbf{x}}_j)\right\})^2} + 2 \cdot \sum_j \hat{p}_j (1 - \mu_j y_j f(\bar{\mathbf{x}}_j))_+ \\
&\quad \textit{subject to} \qquad 0 < \mu_j \leq 1 \quad \forall j. \\
&= \frac{1}{2} \cdot \sum_j \hat{p}_j \frac{\mathbb{E}\,(\mathbf{v}^T(\mathbf{x}_j - \bar{\mathbf{x}}_j))^2}{(\max\left\{\frac{1}{\mu_j}, y_j f(\bar{\mathbf{x}}_j)\right\})^2} + 2 \cdot \sum_j \hat{p}_j (1 - \mu_j y_j f(\bar{\mathbf{x}}_j))_+ \\
&\quad \textit{subject to} \qquad 0 < \mu_j \leq 1 \quad \forall j. \\
&= \frac{1}{2} \cdot \sum_j \frac{\mathbf{v}^T \mathbf{D}_j \mathbf{v}}{(\max\left\{\frac{1}{\mu_j}, y_j f(\bar{\mathbf{x}}_j)\right\})^2} + 2 \cdot \sum_j \hat{p}_j (1 - \mu_j y_j f(\bar{\mathbf{x}}_j))_+ \qquad \text{(A.3)} \\
&\quad \textit{subject to} \qquad 0 < \mu_j \leq 1 \quad \forall j.
\end{aligned}$$

The equation in (A.2) uses the fact that the following equality holds when $\mathbf{x}_j$ is symmetrically distributed about its mean:

$$\mathop{\mathbb{E}}_{j} \left[\left(\frac{f(\mathbf{x}_j) - f(\bar{\mathbf{x}}_j)}{\max\left\{\frac{1}{\mu_j}, y_j f(\bar{\mathbf{x}}_j)\right\}} \right)^2 \mathbf{1}(y_j f(\mathbf{x}_j) \leq y_j f(\bar{\mathbf{x}}_j)) \right] =$$
$$\mathop{\mathbb{E}}_{j} \left(\frac{f(\mathbf{x}_j) - f(\bar{\mathbf{x}}_j)}{\max\left\{\frac{1}{\mu_j}, y_j f(\bar{\mathbf{x}}_j)\right\}} \right)^2 \cdot \mathop{\mathbb{E}}_{j} \mathbf{1}(y_j f(\mathbf{x}_j) \leq y_j f(\bar{\mathbf{x}}_j)).$$

Since $\mathbf{x}_j$ is symmetrically distributed about $\bar{\mathbf{x}}_j$ and since f is an affine function, we know that $f(\mathbf{x}_j)$ is symmetrically distributed about $f(\bar{\mathbf{x}}_j)$. Thus, $\mathbb{E}_j\ \mathbf{1}(y_j f(\mathbf{x}_j) \leq y_j f(\bar{\mathbf{x}}_j)) = \frac{1}{2}$, which explains the origin of the "$\frac{1}{2}$" term in the preceding derivation. We now relax the

bound on $\mathcal{R}(f)$ in the following way:

$$\mathcal{R}(f) \leq \frac{1}{2} \cdot \sum_j \frac{\mathbf{v}^T \mathbf{D}_j \mathbf{v}}{(\max\left\{\frac{1}{\mu_j}, y_j f(\bar{\mathbf{x}}_j)\right\})^2} + 2 \cdot \sum_j \hat{p}_j (1 - \mu_j y_j f(\bar{\mathbf{x}}_j))_+ \tag{A.4}$$
$$\textit{subject to} \qquad 0 < \mu_j \leq 1 \quad \forall j.$$

$$\leq \frac{1}{2} \cdot \mathbf{v}^T (\sum_j \mu_j^2 \mathbf{D}_j)\mathbf{v} + 2 \cdot \sum_j \hat{p}_j \mu_j \xi_j \tag{A.5}$$
$$\textit{subject to} \qquad \xi_j = (\tfrac{1}{\mu_j} - y_j(\mathbf{v}^T \bar{\mathbf{x}}_j + b))_+ \quad \forall j,$$
$$0 < \mu_j \leq 1 \quad \forall j.$$

$$\leq \frac{1}{2} \cdot \mathbf{v}^T (\sum_j \mu_j^2 \mathbf{D}_j)\mathbf{v} + 2 \cdot \sum_j \hat{p}_j \mu_j \xi_j \tag{A.6}$$
$$\frac{1}{\mu_j} - \xi_j \leq y_j(\mathbf{v}^T \bar{\mathbf{x}}_j + b) \quad \forall j,$$
$$\textit{subject to} \qquad 0 \leq \xi_j \quad \forall j,$$
$$0 < \mu_j \leq 1 \quad \forall j.$$

$$\leq \frac{1}{2} \cdot \mathbf{v}^T (\sum_j \mu_j \mathbf{D}_j)\mathbf{v} + 2 \cdot \sum_j \hat{p}_j \xi_j \tag{A.7}$$
$$\frac{1}{\mu_j} - \xi_j \leq y_j(\mathbf{v}^T \bar{\mathbf{x}}_j + b) \quad \forall j,$$
$$\textit{subject to} \qquad 0 \leq \xi_j \quad \forall j,$$
$$0 < \mu_j \leq 1 \quad \forall j.$$

$$\leq \frac{1}{2} \cdot \mathbf{v}^T (\sum_j \mathbf{D}_j)\mathbf{v} + 2 \cdot \sum_j \hat{p}_j \xi_j \tag{A.8}$$
$$1 - \xi_j \leq y_j(\mathbf{v}^T \bar{\mathbf{x}}_j + b) \quad \forall j,$$
$$\textit{subject to} \qquad 0 \leq \xi_j \quad \forall j.$$

Note that the bound in (A.4) is the same as the bound in (A.3). In (A.5), we relax the bound on $\mathcal{R}(f)$ by setting the denominator equal to $\frac{1}{\mu_j^2}$. We further relax the bound by changing the equality constaint on ξ_j in (A.5) to an inequality constraint in (A.6). Since μ_j is constrained to lie in the range $(0, 1]$, we can upper bound (A.6) by changing μ_j^n to μ_j^{n-1}. This gives us the bounds in (A.7) and (A.8). The bound in (A.8) is the same as in Theorem 11 and the bound in (A.7) is the same as in Theorem 13. □