Regularized Adaptation: Theory, Algorithms and Applications

Xiao Li

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

Doctor of Philosophy

University of Washington

2007

Program Authorized to Offer Degree: Electrical Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Xiao Li

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of the Supervisory Committee:

____
Jeff Bilmes

Reading Committee:

____
Jeff Bilmes

____
Katrin Kirchhoff

____
Marina Meila

Date: __________________________

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Proquest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 1-800-521-0600, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature________________________________

Date____________________________________

University of Washington

Abstract

Regularized Adaptation: Theory, Algorithms and Applications

Xiao Li

Chair of the Supervisory Committee:
Professor Jeff Bilmes
Electrical Engineering

Many statistical learning techniques assume that training and testing samples are generated from the same underlying distribution. Often, however, an ”unadapted classifier” is trained on samples drawn from a training distribution that is different from the target (or test-time) distribution. Moreover, in many applications, while there may be essentially an unlimited amount of labeled ”training data,” only a small amount of labeled ”adaptation data” drawn from the target distribution is available. The problem of adaptive learning (or adaptation) then, is to learn a new classifier utilizing the unadapted classifier and the limited adaptation data, in an attempt to obtain as good classification performance on the target distribution as possible.

The goal of this dissertation is to investigate theory, algorithms and applications of adaptive learning. Specifically, we propose a Bayesian “fidelity prior” for classifier adaptation, which leads to simple yet principled adaptation strategies for both generative and discriminative models. In the PAC-Bayesian framework, this prior relates the generalization error bound to the KL-divergence between training and target distributions. Furthermore, based on the fidelity prior, we develop “regularized adaptation” algorithms in particular for support vector machines and multi-layer perceptrons. We evaluate these algorithms on a vowel classification corpus for speaker adaptation, and on an object recognition corpus for lighting condition adaptation. Experiments show that regularized adaptation yielded superior performance compared with other adaptation strategies.

The theoretical and algorithmic work on adaptive learning was originally motivated by the development of the “Vocal Joystick” (VJ), a voice based computer interface for individuals with motor

impairments. The final part of this dissertation describes the VJ engine architecture, with focus on the signal processing and pattern recognition modules. We discuss the application of regularized adaptation algorithms to a vowel classifier and a discrete sound recognizer in the VJ, which greatly helped enhance the engine performance. In addition, we present other machine learning techniques developed for the VJ, including a novel pitch tracking algorithm and an online adaptive filter algorithm.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table Number Page

ACKNOWLEDGMENTS

I am enormously grateful to my advisor Jeff Bilmes for accepting me as a graduate student when I knew nothing about research, and for teaching me everything I know about research. I especially thank him for never letting me give up, and never letting me get away with anything unpolished. I am grateful to three other committee members, Katrin Kirchoff and Marina Meila and Maya Gupta for providing very important comments on my work. And Mari Ostendorf for offering two inspirational courses EE517 and EE596 that have greatly influenced my research interests.

My work would not have been possible without the collaborations from my colleagues Jonathan Malkin and Susumu Harada, who have been working closely with me on the Vocal Joystick project. Jonathan has made key contributions to the Vocal Joystick system, including catching a few critical bugs I created. Susumu has rewritten the VJ engine, in an amazingly short time, with a new software architecture, and has been a terrific source knowledge for user studies. I am also indebted to Kelley Kilanski and Richard Wright for their great efforts in data collection. especially thank Kelley for annotating thousands of recordings – a job that would have driven anyone crazy.

Many thanks to my fellow graduate students in the SSLI Lab. It has been a great pleasure working with them side by side. Thank Gang Ji for his tremendous help in my early years of graduate school. Thank Arindam Mandal, Chris Bartels, Dustin Hillard, Jonathan Malkin, Kevin Duh, Sheila Reynolds for helping me practice for my final defense. Thank Xin Lei, Mei Yang, Karim Filali, Arindam Mandal, Chris Bartels for ordering food together when working late at night and for interesting discussions. Also thank other fellow students, Andrei Alexandrescu, Jeremy Kahn, Mukund Narasimhan, Scott Otterson, Sarah Petersen, Raghu Kumaran and Amar Subramanya, for deciding to stay packed and together in EE1-203 rather than being distributed to separate offices, and for various other things — they have made SSLI a fun and stimulating place to work in.

I would like to express my gratitude to the sources of my financial support. My research was funded in part by the National Science Foundation under Grant IIS-0326382, and later by a Microsoft Research Fellowship. I would especially thank Alex Acero, Asela Guanawardana at Microsoft Research for their support in my fellowship application, the Speech Technologies group for offering me internships.

Thank my dearest friends, Cathy and Christie, for their constant support thousands of miles away; Xiaobei and Claire, for sharing with me their laughters and tears.

I am forever indebted to my parents, who always believed in me, and to Yeping, who made me believe.

Finally, thank the Ph.D. comic strip by Jorge Cham for making me laugh through the toughest moments of graduate school.

”Reasonable people adapt themselves to the world. Unreasonable people attempt to adapt the world to themselves. All progress, therefore, depends on unreasonable people.”

—- *George Bernard Shaw*

Chapter 1
INTRODUCTION

Many concepts and techniques in machine learning are illuminated by human (or animal) learning behaviors [1]. Indeed, human's ability to learn is a hallmark of intelligence. Perhaps the most simple form of human learning is ”trivial” memorization of experience. However, it is amazing how much and how fast humans can memorize; they remember and recognize complicated objects and sounds, such as faces and voices, with seemingly no difficulty even when they have experienced these objects or sounds for only seconds. This makes us wonder what exactly are being memorized — every single detail of an object or a sound, or only some salient characteristics? Humans not only learn by memorization (which might involve extracting the most important characteristics of a matter), but also generalize what they have learned. For example, in a classic experiment by [2], a group of English-speaking children were able to correctly pluralize a novel word that they had never encountered. It seems that humans are able to induce certain forms of ”rules” based on particular examples, though it is mysterious how this induction process works in the brain. Even more amazingly, humans are constantly transferring their knowledge and skills learned in one context to another context [3]. For example, previous experience of learning to speak English may help learn to speak French, and previous experience of learning to play violin may help learn to play piano. And it has been hypothesized that the degree of knowledge transfer between initial and later learning depends on the match between the elements across two tasks [4]. Yet another important characteristic of human learning is the ability to **adapt**, which can be considered as a special case of transfer of learning. Here by ”adaptation” we mean the process of adjusting the knowledge or skills of prior learning to a specific task. Suppose that a tennis player A wants to beat his rival player B (without worrying about other players for the moment). After having practiced with a great number of players to improve his general tennis skill, A may want to adapt his skill toward the goal of beating B by practicing with a few players whose styles are the closest to B.

As computers becomes more and more powerful, the idea that computers can imitate human learning processes is no longer science fiction. In fact, there has been a surge of interest to study machine learning paradigms that parallel human learning processes, such as efficient knowledge representation, induction, and transfer. These techniques have greatly influenced the development of more intelligent computer interfaces that can recognize objects and sounds, understand human languages, predict weather and traffic, diagnose diseases, detect fraudulent financial transactions, or even play chess.

This dissertation is particularly interested in the setting of adaptive learning, which is analogous to the human learning process of adapting knowledge and skills toward a target task. Indeed, many statistical learning techniques assume that training and testing samples are generated from the same underlying distribution. Often, however, an "unadapted classifier" is trained on samples drawn from a training distribution that is not the same as the target (or test-time) distribution. Moreover, in many applications, while there may be essentially an unlimited amount of labeled "training data," only a small amount of labeled "adaptation data" drawn from the target distribution is available. The problem of adaptation, then, is to learn a new classifier utilizing the unadapted classifier and the limited adaptation data, in an attempt to obtain as good classification performance on the target distribution as possible.

Adaptation is most useful in scenarios where a computer interface needs to be customized for a target user or domain, or to be adjusted constantly in an evolutionary environment. In speech and handwriting recognition, for example, an unadapted classifier may be trained on a database consisting of samples from an enormous number of individuals. The target distribution would correspond only to a specific user, from whom it would be unrealistic to obtain a large amount of labeled training data. A system, however, should be able to quickly adapt to that user by combining information from the large corpus and as small an amount of adaptation data as possible. Taking text classification as another example, it is often the case that a vast amount of labeled training data exists in one domain, but the target task is to be conducted in another domain where data annotation is expensive. Domain adaptation, which utilizes information from the original domain, may help reduce the amount of new annotation efforts while achieving good performance in the target domain.

The theoretical and algorithmic work that will be presented in dissertation was originally motivated from the development of the *Vocal Joystick* (VJ). The VJ is a voice based computer interface designed for individuals with motor impairments. Unlike conventional speech interfaces, the Vocal Joystick exploits continuous aspects of human's voice, such as intensity, pitch and vowel quality, and produces continuous signals which can be used to control mouse pointers, robotic arms, or other electro-mechanical devices. Moreover, similar to speech interfaces, the VJ almost always finds its best performance when adapted to the target user, and this is where the work developed in this dissertation comes into play.

This chapter serves as an introduction to this dissertation. Section 1.1 gives a comparative view of a number of machine learning paradigms, including *adaptive learning* which is the essence of this dissertation. Section 1.2 introduces the Vocal Joystick, a real-world application that motivated this work, and states the potential learning and adaptation problems in the VJ system. Finally, the last section summarizes the contributions and presents a roadmap of this dissertation.

1.1 Machine Learning Paradigms

A variety of machine learning paradigms have been studied in the past including *supervised*, *unsupervised*, and *semi-supervised learning*, as well as *transductive* vs. *inductive learning*, and more recently, *multi-task learning* or *transfer learning*, any of which may be seen from either a frequentist or Bayesian perspective. A learning setting that has not received as much theoretical attention is that of *adaptive learning*. This section offers a comparative review of these learning paradigms as well as a brief introduction to adaptive learning. For a better understanding of how these learning paradigms differ from or relate to each other, we view them from several different perspectives.

1.1.1 Supervised, unsupervised and semi-supervised learning

We first inspect the way that training samples are labeled, which essentially makes the distinction between supervised, unsupervised and semi-supervised learning. Supervised learning assumes that *all* training samples are labeled, whereas unsupervised learning assumes *none*. Formally, in the former setting both inputs and their labels $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ are given, while in the latter only inputs $\{\mathbf{x}_i\}_{i=1}^m$ are available. It may be somewhat mysterious what the machine could possibly learn without any labeled data. Unsupervised learning, in fact, often aims at building representations of the input that can be used for prediction, decision making or data compression [5]. For example, density estimation [6], clustering [7, 8], principle component (or surface) analysis [8], independent component analysis [9] are all important forms of unsupervised learning.

Notice that in some cases there exists another label variable k_i, parallel to y_i, that provides information on $\mathbf{x}_i$ at a different level. For example, in Gaussian mixture models, we can have y_i representing the Gaussian mixture ID and k_i representing the component ID in that Gaussian mixture. In this regard, we should distinguish between the “fully supervised” case, where both y_i and k_i are available, the “partially supervised” case, where y_i is available but k_i is not, and the “fully unsupervised” case, where neither y_i nor k_i are present.

Semi-supervised learning, as what “semi” may have suggested, assumes that there exist both labeled training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$ and unlabeled ones $\{\mathbf{x}_i\}_{i=m+1}^{m+n}$. In many machine learning applications, unlabeled data is abundant but labeling is expensive and time-consuming. The basic idea of semi-supervised learning is to use the input distribution learned from the unlabeled data to influence the supervised learning problem [10]. In the probabilistic framework, semi-supervised learning can be treated as a missing data problem, which can be addressed by generative models using the EM algorithm and extensions thereof [11–13]. Self-training [14] extends the idea of EM to a wider range of classification models: it iteratively trains a seed classifier using the labeled data (sometimes with regularization [15]), and uses high-confidence predictions on the unlabeled data to expand the training set. Co-training [16] assumes that the input features can be split into two conditionally independent subsets, and that each subset is sufficient for classification. Under these assumptions, the algorithm trains two separate classifiers on these two subsets of features, and each classifier’s predictions on new unlabeled samples are used to enlarge the training set of the other. This approach often improves over self-training, as compared in [17]. Another school of methods for semi-supervised learning are based on graphs [18], where nodes represent labeled and unlabeled samples, and edges reflect the similarity between the samples. Given such a graph, we desire to find a decision function that satisfies the constraints imposed by the labeled data and is smooth over the entire graph [18–20].

Additionally, *active learning* is a similar setting as semi-supervised learning, but it allows an intelligent choice of which samples to label. For example, query learning [21] or selective sampling [22] generates or selects the most informative inputs for the human expert to label with the hope to improve classification performance with the minimal amount of queries.

1.1.2 Inductive and transductive learning

A second perspective, which distinguishes inductive learning from transductive learning, has to do with the scope in which a classifier is devised to work. Assume that we observe a set of samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ drawn from a distribution $p(\mathbf{x}, y)$ in the sample space $\mathcal{X} \times \mathcal{Y}$. Inductive learning aims to learn a decision function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that not only correctly classifies observed samples, but also generalizes to *any* unseen samples drawn from the same distribution. In other words, we desire to learn an f that minimizes the *expected risk*, *i.e.*, $R_{p(\mathbf{x},y)}(f) = \mathrm{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y)}[Q(f(\mathbf{x}), y)]$, under some loss function $Q(\cdot)$, which will be formally discussed in Chapter 2.

In transductive learning [23], we are further given a set of unlabeled inputs $\{\mathbf{x}_i\}_{i=m+1}^{m+n}$, and we only care about predicting as accurately as possible the labels $\{y_i\}_{i=m+1}^{m+n}$ of these *target* inputs. Since a transduction algorithm does not need to generalize, it is usually devised explicitly to find the right labels instead of to construct a decision function. A transductive SVM [23–25], however, outputs a decision function which can potentially handle unseen data. In fact, a transductive SVM is in the strict sense an inductive learner, although it is by convention called "transductive" for its intention to minimize the generalization error bound on the target inputs [23]. It is worth noting that semi-supervised learning is often mistaken for transductive learning, as both learning paradigms receive partially labeled data for training. In fact, semi-supervised learning can be either inductive or transductive, depending on whether its goal is to generalize.

1.1.3 Meta-learning

Meta-learning, also referred to as multi-task learning or transfer learning, deals with the problem of "learning to learn". This paradigm involves a higher level of generalization. While learning at the base level desires to produce a learner that generalizes across samples drawn from a specific distribution or domain, meta-learning emphasizes producing a meta-learner that generalizes across distributions and domains. Here we discuss meta-learning in the context of inductive learning.

For a better understanding of the problem, we refer to a formulation presented in [26], but we replace the notation therein with our own notation for consistency with the rest of this dissertation. Assume that there are K different yet "related" tasks. Each task defines a distribution $p^k(\mathbf{x}, y)$ on the same sample space $\mathcal{X} \times \mathcal{Y}$, from which a training set $\mathcal{D}^k$ is generated. The tasks relate to each other via a meta-parameter θ, the form of which will be discussed shortly. One setting of meta-learning (mostly referred to as multi-task learning) is concerned with simultaneously learning K specific tasks. The goal is to jointly find a meta-parameter θ and a set of decision functions $f^k \in \mathcal{F}^k$, $k = 1..K$, that minimize the average expected risk, *i.e.*,

$$\min_{\theta, f_1 \in \mathcal{F}^1, .., f_K \in \mathcal{F}^K} \frac{1}{K} \sum_k R_{p^k(\mathbf{x},y)}(f^k),$$

subject to some "relatedness" constraint parameterized by θ. It has been theoretically proved that learning multiple tasks jointly is guaranteed to have better generalization performance than learning them independently, given that these tasks are related [26–30]. A second setting of meta-learning assumes that p^k, $k = 1..K$, are sampled from a "meta-distribution" $q(p)$ [26, 27]. Under such an assumption, the goal is to find a meta-parameter θ that generalizes to unseen sample distributions $p(\mathbf{x}, y)$ drawn from $q(p)$, *i.e.*,

$$\min_{\theta} E_{p \sim q(p)} [\inf_{f \in \mathcal{F}} R_{p(\mathbf{x},y)}(f)],$$

again subject to the relatedness constraint parameterized by θ.

In both cases, the relatedness constraint (parameterized by θ) is the key to the success of meta-learning. There are many ways to define this constraint by exploiting different types of "relatedness". From a frequentist perspective, [26] defines relatedness between tasks as an "internal representation". Specifically, it is assumed that $\mathcal{F}^k = \mathcal{F}_\theta$ for all k, where $\mathcal{F}_\theta$ is chosen from a family of function spaces. For example, $\mathcal{F}_\theta$ can be represented by a space of multi-layer perceptrons where the first two layers, parameterized by θ, are shared by all tasks, while the remaining layers are task-dependent [26, 31]. Another form of relatedness is given by [32] and [30], where the decision function f^k is assumed to be a linear combination of a task-independent and a task-dependent component. Furthermore, [29] defines relatedness between tasks on the basis of similarity between distributions. Formally, two distributions $p^k(\mathbf{x}, y)$ and $p^l(\mathbf{x}, y)$ are related if there exists some transformation $T : \mathcal{X} \to \mathcal{X}$ such that $p^k(\mathbf{x}, y) = p^l(T(\mathbf{x}), y)$. On the other hand, there are various Bayesian approaches to meta-learning, many of which are based on hierarchical Bayesian inference [33]. The constraint therein is that the decision functions f^k from multiple tasks are generated from the same prior distribution $p_\theta(f)$. Empirical study of this approach to multi-task learning can be found in a number of publications such as [34, 35].

Table 1.1: A comparative view of learning paradigms

Paradigms	given	desire to learn
Supervised	$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$	
Unsupervised	$\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^m$	
Semi-supervised	$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \bigcup \{\mathbf{x}_i\}_{i=m+1}^{m+n}$	
Inductive	$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ or $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \bigcup \{\mathbf{x}_i\}_{i=m+1}^{m+n}$ where $(\mathbf{x}_i, y_i) \sim p(\mathbf{x}, y)$	$\underset{f}{\text{argmin}}\ R_{p(\mathbf{x},y)}(f)$
Transductive	$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \bigcup \{\mathbf{x}_i\}_{i=m+1}^{m+n}$	labels of $\{\mathbf{x}_i\}_{i=m+1}^{m+n}$
Meta-learning (setting I)	$\{\mathcal{D}^k\}_{k=1}^K$ (fully or partially labeled) where $(\mathbf{x}_i, y_i) \sim p^k(\mathbf{x}, y)$	$\underset{\theta, f^1, .., f^K}{\text{argmin}}\ \frac{1}{K}\sum_{k=1}^{K} R_{p^k(\mathbf{x},y)}(f^k)$ s.t. "relatedness" constraint by θ
Meta-learning (setting II)	$\{\mathcal{D}^k\}_{k=1}^K$ (fully or partially labeled) where $(\mathbf{x}_i, y_i) \sim p^k(\mathbf{x}, y)$, $p^k \sim q(p)$	$\underset{\theta}{\text{argmin}}\ \mathrm{E}_{f^k \sim q(f)}[\inf_f R_{p^k(\mathbf{x},y)}(f^k)]$ s.t. "relatedness" constraint by θ
Adaptive	$f^{tr} \in \underset{f}{\text{argmin}}\ R_{p^{tr}(\mathbf{x},y)}(f \in \mathcal{F})$ and $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ where $(\mathbf{x}_i, y_i) \sim p^{ad}(\mathbf{x}, y)$	$\underset{f \in \mathcal{F}}{\text{argmin}}\ R_{p^{ad}(\mathbf{x},y)}(f)$

1.1.4 Adaptive learning

Finally, we introduce the idea of adaptive learning (or adaptation); a detailed discussion will be given in Chapter 4. As mentioned at the beginning of this chapter, adaptive learning is concerned with situations where the target sample distribution denoted by $p^{ad}(\mathbf{x}, y)$, deviates from that of training, denoted by $p^{tr}(\mathbf{x}, y)$. In such a situation, no matter how much data is available from the training-data distribution, it will not be possible to obtain an asymptotically consistent estimate. On the other hand, training a model using only the adaptation data would [8] either lead to: (1) overfitting, due to a high-variance parameter estimate of a complex model, or (2) high-bias due to estimating the parameters of an excessively simple model. In this work, we assume the availability of an unadapted decision function f^{tr} learned using a sufficient amount of training-distribution data, as well as some labeled adaptation data $\mathcal{D}^{ad}$ drawn from the target distribution. The goal of adaptation is to learn an optimal decision f^{ad} function with regard to $p^{ad}(\mathbf{x}, y)$. In this paradigm, $\mathcal{D}^{ad}$ are *true* samples from the target distribution, while f^{tr} is a good representative only of the training distribution, which may at best be only similar but not identical to the target distribution. By combining these two sources of information, one would hope to achieve better performance than using either one alone.

Adaptive learning is most akin to multi-task learning in that learning the unadapted model and learning the adapted model can be viewed as two similar tasks, each with a different sample distribution. However, our setting of adaptive learning is not entirely the same as the problem of simultaneously learning multiple tasks. First of all, we are only interested in the performance of the target task rather than the average performance over all tasks. Secondly, the training data is no longer available at adaptation time — the only information preserved from training is the unadapted model. Note that the second assumption is fairly common in real-world scenarios when it is unrealistic to deliver a large amount of training data to end users.

These learning paradigms are summarized in Table 1.1, with a few points that we need to clarify: (1) The first four learning paradigms differ from each other only by "what is given", regardless of "what they desire to learn" (thus we leave the corresponding entries empty); (2) We discussed meta-learning and adaptive learning in the context of inductive learning, but both can be transductive. (3) The notation $R_{p(\mathbf{x},y)}(f)$ denotes expect risk w.r.t. the sample distribution $p(\mathbf{x}, y)$, which will be formally introduced in the next Chapter.

1.2 The Vocal Joystick

While this dissertation studies the problem of adaptation primarily from a machine learning perspective, the incentive of conducting such research was originated from the development of the Vocal Joystick (VJ) — a computer interface for individuals with motor impairments. This section presents the motivation and background of the VJ project. Notice that much of the text in this section was written on the basis of a number of seminal publications by Bilmes and et. al. [36,37]. Furthermore, the development of the Vocal Joystick was a joint work with a number of faculties and students at the University of Washington. We will acknowledge their work in detail in Section 1.4

As human-computer interaction becomes ubiquitous, the concept of continuous interaction [38] increasingly impacts the way we interact with everyday objects and appliances, and ultimately on the way we live in the modern world. Many existing interfaces such as mouse pointers, joysticks and touch-pads, however, are ill-suited for individual with motor impairments. There has been a sustained interest in the computer-human interaction (CHI) society to develop hand-free computer interfaces for this population. A sip-and-puff switch, for example, is a head-mounted accessory used to actuate a binary-mode switch by a simple sip or puff [39, 40]. This is often used in company with a head-mouse, or a chin-joystick, which controls a standard computer mouse or joystick by measuring the user's head or chin movements [39, 40]. Such interfaces remove the constraints on the user's ability to use hands, but often suffer from low communication bandwidths. An eye tracker provides an attractive alternative that maps eye-gaze positions to mouse pointer positions, but it often requires expensive hardware and elaborate calibration efforts before use [41].

A speech interface is a more natural solution, as speech is a major facility in human-human communication. Such an interface recognizes and understands speech inputs using an automatic speech recognizer and launches actions accordingly [42, 43]. Standard spoken language commands, however, are most useful for discrete but not for continuous operations, and thus are ill-suited for manipulating computers, windows-icons-mouse-pointer (WIMP) interfaces and other electro-mechanical devices. For example, in order to move a cursor from the bottom-left to the upper-right of a screen, a user would have to repeatedly utter "up" and "right", or "stop" and "go" after setting an initial movement direction and speed. An alternative strategy would be to verbalize the intent in a more sophisticated manner, *e.g.*, "move mouse two o'clock" [43], but this would increase the cognitive load of the user by imposing syntax and semantic restrictions. More importantly, both strategies suffer from the discontinuous nature of discrete command control and the inability to control the movement speed efficiently.

Therefore, it is useful to develop a continuous voiced-based interface such that the computer does not need to wait for a complete command to actuate an operation, but rather continuously listens to the user and responds quietly to his/her interests. A number of systems have been proposed for this purpose in the CHI community, among which [44] and [45] are the most similar to the VJ. Both systems have utilized non-verbal acoustic parameters, such as duration and pitch of vocalization, for continuous control of mouse pointers. After the movement direction is initialized by a discrete command, a movement can be launched by any vowel articulation, and the movement continues until the vowel articulation stops; in the meantime, the movement speed can be changed on-the-fly by varying pitch [44]. The systems proposed by [43, 46, 47] work in a similar fashion, except that they do not require continuous vowel articulation; once the movement direction has been set, the mouse pointer moves automatically until a discrete command is received to stop the movement. All these techniques, however, lack the ability to fluidly and continuously change the movement direction and, in some cases, the movement speed without having to issue discrete commands.

The Vocal Joystick project, funded by NSF and conducted at the University of Washington, aims to assist individuals with motor impairments using a more fluid control mechanism. This voice-based interface is not confined to the use of natural languages, but exhaustively explores non-verbal acoustic parameters of human voice such as intensity, pitch, vowel quality and discrete sounds (comprised of phoneme combinations). We have utilized this interface to control mouse movement using the following scheme: mouse movement is triggered when vowel activity is detected, and continues until the vowel activity stops. At each time frame, the movement direction is determined by vowel quality, while the step size is determined by loudness. Finally, a small set of discrete sounds are used to execute mouse clicks. A more detailed description of the Vocal Joystick will be given in Chapter 6.

The Vocal Joystick has the following advantages compared with speech interfaces: (1) The VJ offers control mechanisms for both continuous and discrete tasks; (2) it can select the vocal parameters that maximally reduce confusability ; (3) it provides instantaneous feedback to a user so that the interaction is mutually adaptive; and (4) it reduces the user's cognitive load. Compared with other continuous interfaces such as a head-mouse, a chin joystick or an eye tracker, the VJ has a relatively high bandwidth, thereby providing more degrees of freedom in control. It is also relatively cheap and requires less efforts in setup.

The design of the VJ system involves a variety of research areas, including phonetics, cognition and perception science, signal processing, machine learning, control, user interface design and usability. This dissertation focuses on several research problems at the pattern recognition level. First, it is crucial for the VJ system to have a robust estimation of the vocal parameters, including loudness, pitch, vowel quality, and discrete sound identity, and to develop an efficient user adaptation scheme to improve classification and estimation performance. Secondly, it is sometimes necessary to have an intelligent filtering algorithm applied to some of the vocal parameters, as it is often difficult for a user to make the precise articulation he/she has intended, For example, in order to move along a non-cardinal direction, the user may repeatedly alter between two cardinal vowels. A second reason is that there often exist system errors in classification; an adaptive filter would greatly help infer the true user intent from noisy estimates.

1.3 Contributions and Organization

This dissertation makes contributions to the problem of adaptive learning from the following three aspects.

- The theoretical contribution of this work is that we present a Bayesian "fidelity prior" for classifier adaptation. This prior unifies the adaptation strategies for a variety of generative models and conditional models, and relates the generalization error bound (or sample complexity bound) to the KL-divergence between training and target distributions in the PAC-Bayesian setting.

- The algorithmic contribution comes from the development of novel regularized adaptation algorithms for SVMs and MLPs, both derived from an lower bound of the fidelity prior. These algorithms perform remarkably well on a vowel classification dataset and an object recognition dataset compared to other state-of-the-art techniques in the literature. Note that we have also implemented an regularized adaptation algorithm for GMMs, which is essentially the same as the conventional MAP adaptation.

- Application-wise, this dissertation gives an overview of the development of the Vocal Joystick system. The main research contribution to the VJ system from this dissertation is the exploitation of machine learning techniques at the signal processing and pattern recognition level. This is specifically concerned with building regularized adaptation tools for the vowel classifier and for the discrete sound recognizer in the VJ engine. In addition, this work describes a novel pitch tracker based on graphical models, as well as an intelligent adaptive filter that compensates for human and system errors.

The rest of the dissertation is organized as follows [1]: Chapter 2 reviews theoretical foundations based on which this work is constructed. Chapter 3 reviews practical work on adaptation developed in the area of speech recognition and natural language processing. Chapter 4 presents the fidelity prior and its instantiations for generative models and discriminative models, and provides PAC-Bayesian error bound analysis. Chapter 5 describes in detail regularized adaptation algorithms for SVMs and MLPs with experiments on two pattern classification tasks. Chapter 6 introduces the Vocal Joystick systems and the adaptation tools. Chapter 7 discusses other machine learning techniques in the VJ system, including a pitch tracking algorithm and an online adaptive filtering algorithm. Finally, Chapter 8 concludes and suggests directions for future work. In addition, non-important proofs (which are trivially derived from others' work) and algorithm derivations can be found in the appendices.

1.4 Collaborations and Publications

Some of the work described in this dissertation has been published in conference proceedings. The fidelity prior and the error bound analysis were published in [48], coauthored with Jeff Bilmes. The regularized adaptation algorithm for MLPs, with evaluation on vowel classification, was first proposed in [49], coauthored with Jeff Bilmes. An empirical design of SVM adaptation was presented in [50], coauthored with Jonathan Malkin and Jeff Bilmes. The pitch tracking algorithm was published in [51], coauthored with Jonathan Malkin and Jeff Bilmes. The online adaptive filter algorithm was published in [52], coauthored with Jonathan Malkin, Susumu Harada, Jeff Bilmes, Richard Wright and James Landay.

As mentioned earlier, the development of the Vocal Joystick involved efforts from many other researchers (faculties and students), and covers a broad spectrum of research areas. The first paper [36] that offered a comprehensive description of the VJ was a joint work by Jeff Bilmes, Xiao Li, Jonathan Malkin, Kelley Kilanski, Richard Wright, Katrin Kirchhoff, Amarnag Subramanya, Susumu Harada, James Landay, Patricia Dowden and Howard Chizeck. A sister paper [37] by the same authors presented engineering details of the VJ system. Research on the motion control module was primarily done by Jonathan Malkin, and was partially published in [53]. Research on the

[1] As a side note, this dissertation is in fact organized in inverse chronological order of the progress of my Ph.D. work.

VJ user interface was primarily done by Susumu Harada, and was published in [54]. Furthermore, research on data collection efforts can be found in [55], first authored by Kelley Kilanski. Finally, the implementation of the VJ engine was a joint work with Jonathan Malkin and Susumu Harada. The graphical user interface for adaptation presented in Figure 6.4 was primarily designed by Susumu Harada.

Chapter 2
LEARNING THEORY FOUNDATIONS

As introduced in Chapter 1, inductive learning is a machine learning paradigm that, given examples of inputs and corresponding outputs, learns to predict outputs on future inputs. The prediction task is called *classification* when we predict qualitative outputs, and *regression* when we predict quantitative outputs. Formally speaking, we assume that there exists a mapping from an input space $\mathcal{X}$ to an output space $\mathcal{Y}$, where $\mathcal{Y}$ is a set of class labels in the case of classification, and a space of real values in the case of regression. Inductive learning asks to discover this mapping (both function structure and parameter values) based on a set of observed examples. This dissertation is concerned with classification tasks in the context of inductive learning.

Throughout this work, all densities are taken w.r.t. the Lebesgue measure in their respective spaces. We use $p(x)$ as a short-cut for $p_X(X = x)$, and $\mathrm{E}_{p(x)}[\cdot]$ as a short-cut for $\mathrm{E}_{X \sim p_X(X)}[\cdot]$. We also introduce the notion of *convergence in probability*. We say that X_n converges to X in probability, denoted by $X_m \xrightarrow{P} X$, if for any $\epsilon > 0$ and for any $\delta > 0$ there exists a natural number M such that for all $m > M$,

$$\Pr\{|X_m - X| < \epsilon\} > 1 - \delta.$$

Assume that we observe m samples $\mathcal{D}_m = \{(\mathbf{x}_i, y_i) | (\mathbf{x}_i, y_i) \sim p(\mathbf{x}, y)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathcal{X}$ are input features and $y_i \in \mathcal{Y}$ are class labels. Further assume that we are given a function space $\mathcal{F}$ (either countable of uncountable) that contains some mappings $f : \mathcal{X} \to \mathcal{Y}$ which we call decision functions or classifiers. Our goal is to learn a decision function (or a classifier) $f \in \mathcal{F} : \mathcal{X} \to \mathcal{Y}$ that not only correctly classifies the observed samples, but also generalizes to unseen samples drawn from the *same* distribution. In other words, we desire to learn an decision function that minimizes the *expected risk* (or, equivalently, the *true risk*),

$$R_{p(\mathbf{x},y)}(f) \triangleq \mathrm{E}_{p(\mathbf{x},y)}[Q(f(\mathbf{x}), y)], \tag{2.1}$$

where $Q(\cdot)$ is a loss function that measures the “cost” of a classification decision. Notice that the difference between the expected risk of f and the *Bayes risk*, *i.e.* $\inf_f R_{p(\mathbf{x},y)}(f)$, consists of two error terms:

$$R_{p(\mathbf{x},y)}(f) - \inf_f R_{p(\mathbf{x},y)}(f) = \left(R_{p(\mathbf{x},y)}(f) - \inf_{f \in \mathcal{F}} R_{p(\mathbf{x},y)}(f) \right) + \left(\inf_{f \in \mathcal{F}} R_{p(\mathbf{x},y)}(f) - \inf_f R_{p(\mathbf{x},y)}(f) \right) \tag{2.2}$$

where the first term on the right hand side is called the *estimation error*, and the second the *approximation error*. Moreover, throughout this work, we use the 0-1 loss, *i.e.* $Q((f(\mathbf{x}), y) = \mathrm{I}(f(\mathbf{x}) \neq y)$, in *evaluating* the classification performance. In this case the expected risk is the true classification error rate. The 0-1 loss, however, is often computationally intractable as an optimization function [56]; surrogates of the 0-1 loss are typically used in actually *training* a classifier, which will be discussed in Section 2.2 in detail.

Regardless of what loss function to use, the expected risk is hard to optimize directly as $p(\mathbf{x}, y)$ is generally unknown. In practice, we often aim to find a decision function $\hat{f}$ that minimizes the *empirical risk* which is defined as

$$R_{\text{emp}}(f) \triangleq \frac{1}{m} \sum_{i=1}^{m} Q(f(\mathbf{x}_i), y_i),\ (\mathbf{x}_i, y_i) \in \mathcal{D}_m. \tag{2.3}$$

In other words, $\hat{f}$ minimizes the average risk over a specific data set $\mathcal{D}_m$. It has been shown that if $Q(\cdot)$ satisfies certain property, then $R_{\text{emp}}(f) \xrightarrow{P} R_{p(\mathbf{x},y)}(f)$ [57]. Furthermore, if $\mathcal{F}$ has a finite VC dimension (to be defined in Section 2.3), then $R(\hat{f}) \xrightarrow{P} \inf_{f \in \mathcal{F}} (f)$ [23, 58]. The sample size m, however, is almost always far from sufficient. Such cases require the use of certain regularization strategy to guarantee good generalization performance [23, 58–61]. In fact, a fundamental problem in machine learning is to bound the expected risk from above by the empirical risk plus a capacity term. This class of bounds are called *generalization error bounds* or *sample complexity bounds*, which provide theoretical justifications for regularization.

This chapter reviews the learning theory foundations on the basis of which our adaptation theorems and algorithms are developed. The organization of this chapter is as follows: Section 2.1 introduces different classification models, *i.e.*, different ways of formulating the decision function $f(\mathbf{x})$. Section 2.2, without considering generalization ability yet, introduces a number of loss functions $Q(\cdot)$ for parameter estimation. Section 2.3, the most crucial to this work, reviews two schools of theory on generalization error bounds. Section 2.4 discusses the role of regularization in machine learning. The last section gives mathematical basics that are necessary to understanding the rest of the work.

2.1 Statistical Models for Classification

In this section, we discuss different formulations of the decision function f by using different statistical models; how to estimate the parameters of these models (*e.g.*, maximum likelihood estimation or discriminative training) is an orthogonal issue which will be discussed in the next section. In particular, we introduce two important approaches: generative models and discriminative models.

2.1.1 Generative models

In this case, the function space $\mathcal{F}$ consists of models that describe sample distributions. A decision function (or a classifier), therefore, is represented by a generative model, and the output of the decision function is the class label that produces the highest joint probability:

$$\text{“decision”} = \underset{y}{\operatorname{argmax}}\ \ln p(\mathbf{x}, y|f) = \underset{y}{\operatorname{argmax}}\ \ln p(\mathbf{x}|y, f)p(y|f). \tag{2.4}$$

Since the decision function is uniquely determined by the generative model, we slightly abuse notation to let f denote a generative model or its parameters, instead of a decision function.

Well-known examples of this approach include Bayesian networks and Markov random fields [62], while the parametric form of $p(\mathbf{x}, y|f)$ varies depending on specific models. A Bayesian network is a directed acyclic graph with vertices representing variables and edges representing dependence relations among the variables. The joint distribution of all variables factorizes over variables given their parents, *i.e.* $p(\mathbf{x}_{1:n}) = \prod_{i=1}^{n} p(\mathbf{x}_i|\,\text{parents}(\mathbf{x}_i))$. By having fewer edges in the graph, the network makes stronger conditional independence assumptions and the resulting model has fewer degrees of freedom. For example, Naïve Bayes and hidden Markov models (HMMs), with simple graph structures, are among the most popularly used statistical classifiers. A Markov random field is similar to a Bayesian network except that it is an undirected graph, thereby capable of representing certain distributions that a Bayesian network can not represent. In this case, the joint distribution of the variables is the product of potential functions over cliques (the maximal fully-connected subgraphs). Formally, $p(\mathbf{x}_{1:n}) = \frac{1}{Z}\prod_{k} \phi_k(\mathbf{x}_{\{k\}})$, where $\phi_k(\mathbf{x}_{\{k\}})$ is the potential function for clique

k, and Z is a normalization constant. Again, the graph structure has a strong relation to the model complexity.

Generative models have long been used in speech, text, vision and bioinformatics for their ability to handle variable-length structured data. For example, a HMM with a fixed set of parameters can generate an observation sequence of arbitrary length. Moreover, generative models has principled ways to treat latent variables, typically using Expectation-Maximization (EM) [63, 64]. Despite these advantages, this approach is in general sub-optimal for classification tasks, as it intends to solve a more difficult density estimation problem rather than to optimize directly for classification performance. One popular method to compensate for this problem is *discriminative training* of generative model parameters [65–70], as will be discussed in the next section. Alternatively, *structural discriminability* is a structure learning approach that learns dependency relationships between random variables (of a generative model) in a discriminative fashion [71, 72].

2.1.2 Discriminative models

Discriminative models, on the other hand, directly model the conditional relationship of labels given input features. One class of discriminative models represent this conditional relationship probabilistically by modeling the conditional probability $p(y|\mathbf{x}, f)$ (here we let f denote a conditional model or its parameters). The output of a decision function is the class label that yields the highest conditional probability, *i.e.*

$$\text{“decision”} = \underset{y}{\text{argmax}}\ \ln p(y|\mathbf{x}, f) \tag{2.5}$$

Log linear models, multi-layer perceptrons (MLPs), conditional maximum entropy (MaxEnt) models and conditional random fields (CRFs) all belong to probabilistic discriminative models. In fact, the *last layers* of MLPs, MaxEnts and CRFs all can be considered as generalized log linear models. In binary classification, a generalized log linear model assumes that

$$\log \frac{p(y = +1|\mathbf{x}, f)}{p(y = -1|\mathbf{x}, f)} = \mathbf{w}^T \phi(\mathbf{x}) + b.$$

where $\phi(\cdot)$ represents a nonlinear transformation from the input space to a feature space Consequently, the conditional distributions are in the form of

$$p(y|\mathbf{x}, f) = \frac{1}{Z} \exp(y(\mathbf{w}^T \phi(\mathbf{x}) + b)).$$

where Z is a normalization constant.

A second class of discriminative models directly model the decision boundary which is the most relevant to classification. An affine decision function, for example, assumes that the decision boundary is a hyperplane. Moreover, if applied in a transformed feature space, this approach can as well model nonlinear decision boundaries. In such cases,

$$\text{“decision”} = \text{sgn}\left(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b\right) \tag{2.6}$$

where $\mathbf{w}$ and b are affine parameters, and $\phi(\cdot)$ is again a nonlinear transformation. Given a fixed $\phi(\cdot)$, f is represented $(\mathbf{w}, b)$. Non-probabilistic discriminative models include kernel methods such as support vector machines (SVMs) and nearest neighbor methods, many of which, however, are endowed with probabilistic interpretations.

Discriminative classifiers directly relate to classification and therefore often yield superior performance. There has been research on combining the advantages of generative and discriminative models in an attempt to cope with structured data and latent variables while maintaining good classification performance. In addition to methods such as discriminative training and structural discriminability that we have mentioned, there are approaches that exploit generative models in discriminative classifiers, *e.g.* the use of the Fisher kernel [73].

2.2 Loss Functions for Parameter Estimation

As mentioned before, a risk function using the 0-1 loss corresponds to classification error rate, which is what we typically use in evaluating a classifier. Ideally, the objective of training should be consistent with that of evaluation, thus we should choose to use the 0-1 loss in training so that we are minimizing exactly the classification error rate. However, this procedure is computationally intractable for many classes of models; surrogates of 0-1 loss are often used instead in order to analytically solve the optimization problem. Here we introduce a number of loss functions, as well as their associated training objectives. The first group of loss functions are based on probabilistic models, while the second group are based on the notion of *margin*. For a thorough study of loss functions and risk bounds, refer to [57]. Since this section is concerned with parameter estimation, f herein denotes model parameters.

2.2.1 Probability-based loss

Historically, there has always been a strong association between a statistical model and its training objective [74]. *Joint likelihood loss*[1], defined as

$$Q(f(\mathbf{x}), y) = -\ln p(\mathbf{x}, y|f), \tag{2.7}$$

is naturally, and most commonly, used by generative models such as Naïve Bayes and HMMs. It is worth noting that other loss functions can be applied to estimating generative models as well, as will be discussed shortly. The resulting training objective using the joint likelihood loss is $\max\limits_f \frac{1}{m}\sum\limits_{i=1}^{m} \ln p(\mathbf{x}_i, y_i|f)$, which is generally known as *maximum likelihood estimation* (MLE) [67]. MLE aims to find parameters that best describe the data, and is statistically consistent under the assumptions that the model structure is correct, that the training data is generated from the true distribution, and that we have an infinite amount of such training data. Moreover, in generative models, the joint likelihood function can often be decomposed into independent sub-problems which can be optimized separately.

The major flaw of using the joint likelihood loss is that the model structure we choose can be wrong, and the training data is rarely sufficient. Since ultimately we desire good classification performance, we should ideally design a loss function to achieve this goal. To this end, *conditional likelihood loss*, defined as

$$Q(f(\mathbf{x}), y) = -\ln p(y|\mathbf{x}, f), \tag{2.8}$$

directly evaluates a model's predication performance. The resulting training objective becomes $\max\limits_f \frac{1}{m}\sum\limits_{i=1}^{m} \ln p(y_i|\mathbf{x}_i, f)$. Typically we use this objective when training discriminative probabilistic models, such as MLPs, MaxEnt models and CRFs. In such cases, the training is also referred to as "maximum likelihood estimation", though the "likelihood" here actually means "conditional likelihood" rather than "joint likelihood". Notice that we can also apply this loss function when training generative models, where we maximize $\frac{1}{m}\sum\limits_{i=1}^{m}\Big(p(\mathbf{x}_i, y_i|f) / \sum\limits_{y} p(\mathbf{x}_i, y|f)\Big)$, leading to a discriminative training technique termed *maximum mutual information estimation* (MMIE) [65, 67,

[1]We use the term "joint likelihood" in order to distinguish from the "conditional likelihood" which will be introduced shortly. Moreover, the loss is actually the negative logarithm of the joint likelihood, where we omit the adjectives for simplicity

75]. Usually there is no globally optimal solution to this objective; stochastic gradient descent [76], or, in some cases, the extended Baulm-Welch algorithm [77] can be utilized to find a local optimum.

Although MMIE demonstrates significant performance advantages over the traditional MLE approach [66], it aims at fitting class posterior distributions rather than directly minimizing error rate. The *minimum classification error* (MCE) method uses a loss function that are more consistent with error rate minimization [66, 68], *e.g.*,

$$Q(f(\mathbf{x}), y) = \sigma\Big(- g_y(\mathbf{x}, f) + \ln\Big[\frac{1}{|\mathcal{Y}|}\sum_{y' \neq y} \exp\{g_y(\mathbf{x}, f)\eta\}\Big]^{1/\eta}\Big) \tag{2.9}$$

where $g_y(\mathbf{x}, f) = \ln p(y|\mathbf{x}, f)$, and $\sigma(\cdot)$ is a sigmoid function that yields a smooth loss function. The intuition behind is that the prediction will be correct as long as $g_y(\mathbf{x}, f)$, in which y is the true class label, ranks the highest among all $g_{y'}(\mathbf{x})$, $y' \in \mathcal{Y}$; and we can achieve this by maximizing the difference, from one side, between $g_y(\mathbf{x}, f)$ and its opponents.

2.2.2 *Margin-based loss*

Margin-based loss functions follow a general expression $Q(f(\mathbf{x}), y) = \tilde{Q}(yf(\mathbf{x}))$, where $yf(\mathbf{x})$ is known as the *margin*. It is easy to see that for binary classification, $Q(f(\mathbf{x}), y) = \mathrm{I}(yf(\mathbf{x}) > 0)$ is equivalent to the 0-1 loss, the minimization of which is NP-hard for many non-trivial classes of functions [56]. We can, however, replace the indicator function with a convex function $\tilde{Q}(\cdot)$, leading to convex surrogates of the 0-1 loss. The main advantage of this is to simultaneously deal with computationally feasible algorithms and to avoid overfitting [57]. Indeed, many machine learning algorithms adopt such an approach. For example, *hinge loss* used in SVM [23, 78], *logistic loss* used in logistic regression [8] and *exponential loss* used in boosting [79, 80] are all in the form of $\tilde{Q}(yf(\mathbf{x}))$, where their respective $\tilde{Q}(\cdot)$ are convex surrogates of the indicator function as shown in Figure 2.1. Next we introduce several such examples.

First, the hinge loss is defined as

$$Q(f(\mathbf{x}), y) = |1 - yf(\mathbf{x}), 0|_+ \tag{2.10}$$

It is easy to see that the hinge loss is convex but not differentiable, and constrained optimization is hence required to solve the optimization problem. This loss is historically used in training SVMs,

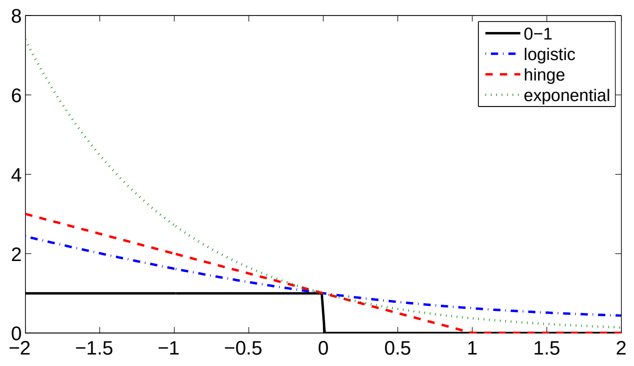


Figure 2.1: Convex surrogates of the 0-1 loss

and is recently utilized in discriminative training of structured generative models such as Markov networks [69] and Gaussian mixture models [70].

Secondly, the logistic loss is defined as

$$Q(f(\mathbf{x}), y) = \ln \frac{1}{1 + \exp(-yf(\mathbf{x}))} \tag{2.11}$$

This coincides with the form of the conditional likelihood loss when the conditional probability $p(y|\mathbf{x}, f)$ uses a sigmoid function. Notice that in Figure 2.1, we actually use a scaled version of the logistic function so that the function value equals one at the point zero.

The exponential loss commonly used in boosting [79, 80] is given by

$$Q(f(\mathbf{x}), y) = \exp\{-yf(\mathbf{x})\} \tag{2.12}$$

Note that it is important to inspect whether $R_{\text{emp}}(f)$ converges to $R_{p(\mathbf{x},y)}(f)$ with sufficiently large m under these surrogate loss functions; and to find quantitative relationships between the estimation error $R_{p(\mathbf{x},y)}(f) - \inf_{f \in \mathcal{F}} R_{p(\mathbf{x},y)}(f)$ associated with the surrogate loss functions and that associated with the 0-1 loss. These issues have been comprehensively investigated in (to name a few) [57, 81, 82].

2.3 Generalization Error Bounds

Recall that in the standard setting of inductive learning, we desire to learn $f^* \in \underset{f \in \mathcal{F}}{\operatorname{argmin}}\ R_{p(\mathbf{x},y)}(f)$. As $p(\mathbf{x}, y)$ is generally unknown, we can instead minimize the empirical risk $\hat{f} \in \underset{f \in \mathcal{F}}{\operatorname{argmin}}\ R_{\text{emp}}(f)$ on a training set $\mathcal{D}_m$. A fundamental question in machine learning is to ask if empirical risk minimization (ERM) is *consistent*, *i.e.*, whether

$$R_{p(\mathbf{x},y)}(\hat{f}) \xrightarrow{P} \inf_{f \in \mathcal{F}} R_{p(\mathbf{x},y)}(f) \tag{2.13}$$

This gives rise to the notion of *strict consistency of empirical risk minimization.* Here we refrain from giving its formal definition as presented in [23], as this would deviate our discussion away from the main theme. Instead, we present a theorem which provides a sufficient *and* necessary condition for strict consistency of ERM, and we present this in our own notation.

Theorem 2.3.1 (Key theorem of learning theory [23]) *Let there exist the constant a and A such that for all functions $f \in \mathcal{F}$, and for a given distribution $p(\mathbf{x}, y)$, the inequality $a \leq R_{p(\mathbf{x},y)}(f) \leq A$ holds true. Then the strict consistency of the ERM principle is equivalent to the uniform one-sided convergence of $R_{emp}(f)$ to $R_{p(\mathbf{x},y)}(f)$,* i.e.*, for any $\epsilon > 0$ and for any $\delta > 0$ there exists a natural M such that for all $m > M$,*

$$\Pr\left\{ R_{p(\mathbf{x},y)}(f) - R_{emp}(f) < \epsilon \right\} > 1 - \delta \tag{2.14}$$

holds true for **all** *$f \in \mathcal{F}$.*

This theorem equates strict consistency with uniform one-sided convergence, and provides a simpler (and sufficient) way to gauge the learning performance which is to use the (ϵ, δ)-bound as shown in Equation (2.14). The (ϵ, δ)-bound is also referred to as a *generalization error bound.*

Here we particularly assume the 0-1 loss function, *i.e.*,

$$\begin{aligned} R_{p(\mathbf{x},y)}(f) &= \mathrm{E}_{p(\mathbf{x},y)}[\mathrm{I}(f(\mathbf{x}) \neq y)] \\ R_{\text{emp}}(f) &= \sum_{i=1}^{m} \mathrm{I}(f(\mathbf{x}_i) \neq y_i) \end{aligned} \tag{2.15}$$

Since $R_{p(\mathbf{x},y)}$ is finite, Theorem 2.3.1 is satisfied. Moreover, we desire a small $R_{\text{emp}}(f)$ as well as small ϵ and δ values, so that we can guarantee a small upper bound on $R_{p(\mathbf{x},y)}(f)$. It has been proven that the following inequality holds true for $f \in \mathcal{F}$,

$$\Pr\left\{ R_{p(\mathbf{x},y)}(f) \leq R_{\text{emp}}(f) + \Phi(\mathcal{F}, f, \mathcal{D}_m, m, \delta) \right\} > 1 - \delta \tag{2.16}$$

where $\Phi(\mathcal{F}, f, \mathcal{D}_m, m, \delta)$ is a capacity term indicating the generalization performance of f (the lower the capacity the better the generalization performance). In general, larger m and δ each lead to smaller $\Phi(\cdot)$, while larger $\mathcal{F}$ (in terms of some capacity measure which will be defined shortly) leads to larger $\Phi(\cdot)$. There is a tradeoff inherited in the generalization error bound (2.16): as the function space $\mathcal{F}$ expands, the empirical error rate $R_{\text{emp}}(f)$ decreases but the capacity term $\Phi(\cdot)$ may at the same time increase, meaning that the decision function f fits the training data better but this is more likely an overfit. An alternative and sometimes equivalent way to measure the generalization performance of a learning algorithm is to inspect the *sample complexity* of achieving an (ϵ, δ)-bound, which is the minimum number of samples needed such that the bound (2.16) holds true; and the upper bound on this number is often referred to as a *sample complexity bound*.

There has been a surge of interest in discovering different forms of the capacity term. In the following two subsections we introduce VC (Vapnik-Cervonenkis) bounds [23] and PAC-Bayesian bounds [83] respectively.

2.3.1 VC bounds

We begin by introducing several important measures describing the capacity of a function space $\mathcal{F}$. Here $\mathcal{F}$ can be either countable or uncountable. Here we assume that $\mathcal{Y} = \pm 1$, *i.e.* the output of $f(\cdot)$ is binary.

Definition $\mathcal{N}(\mathcal{F}, \mathcal{D}_m)$ is defined to be the cardinality of the maximum set of functions $f \in \mathcal{F}$ which yield *different* outputs $[f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_m)]^T$ (each output is a vector of m binary elements) when restricted to a sample set $\mathcal{D}_m$.

This quantity measures the capacity of a function space $\mathcal{F}$ with respect to a particular sample set. More accurately, it measures the number of ways that a function class can separate a specific sample set into two classes. It is easy to see that $\mathcal{N}(\mathcal{F}, \mathcal{D}_m)$ is a non-decreasing function of m, and that $\mathcal{N}(\mathcal{F}, \mathcal{D}_m) \leq 2^m$ since m samples can yield at most 2^m different outputs. Furthermore, this quantity depends on the choice of $\mathcal{D}_m$. For example, consider a set of $m = 3$ samples in the input space $\mathcal{X} = R^2$, and assume $\mathcal{F}$ consists of affine functions. If $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are *not* collinear (*i.e.* they do not lie on a single line), then $\mathcal{N}(\mathcal{F}, \mathcal{D}_3)$ is $2^3 = 8$. If, however, $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are collinear, $\mathcal{N}(\mathcal{F}, \mathcal{D}_3)$ would be at most 6.

The concepts of the *VC entropy* and the *annealed entropy*, defined on the basis of $\mathcal{N}(\mathcal{F}, \mathcal{D}_m)$, are measures of the "expected capacity" of a function space, which no longer depend on a specific sample set $\mathcal{D}_m$.

Definition *VC entropy* $H_{\mathcal{F}}(m) \triangleq E_{p(\mathbf{x},y)}[\ln \mathcal{N}(\mathcal{F}, \mathcal{D}_m)]$

Definition *Annealed entropy* $H^{\text{ann}}_{\mathcal{F}}(m) \triangleq \ln E_{p(\mathbf{x},y)}[\mathcal{N}(\mathcal{F}, \mathcal{D}_m)]$

These two concepts, however, are distribution-dependent, which are hard to evaluate since the sample distribution is usually unknown. The *growth function* is a capacity concept independent of the sample distribution by taking a supremum over all samples.

Definition *Growth function* $G_{\mathcal{F}}(m) \triangleq \ln \sup_{\mathcal{D}_m \in \mathcal{X}^m \times \mathcal{Y}^m} \mathcal{N}(\mathcal{F}, \mathcal{D}_m)$.

Notice that $\sup_{\mathcal{D}_m \in \mathcal{X}^m \times \mathcal{Y}^m} \mathcal{N}(\mathcal{F}, \mathcal{D}_m)$ is often called the *shattering coefficient*, and the growth function is simply its logarithm. It is easy to prove that

$$H_{\mathcal{F}}(m) \leq H^{\text{ann}}_{\mathcal{F}}(m) \leq G_{\mathcal{F}}(m) \tag{2.17}$$

where the first inequality immediately follows Jensen's inequality [84], while the second inequality is obvious. These concepts provide a quantitative measure (though hard to compute) for the function space $\mathcal{F}$. In fact, Vapnik derived different capacity terms $\Phi(\cdot)$ in Equation (2.16) using these measures. Generally, given fixed m and δ, $\Phi(\cdot)$ is a monotonically increasing function of $H_{\mathcal{F}}(m)$, $H^{\text{ann}}_{\mathcal{F}}(m)$, or $G_{\mathcal{F}}(m)$.

Next we briefly introduce the "three milestones"of learning theory [23].

1. A sufficient condition for consistency of the ERM principle is $\lim_{m\to\infty} \frac{H_{\mathcal{F}}(m)}{m} = 0$
2. A sufficient condition for a fast rate of convergence is $\lim_{m\to\infty} \frac{H^{\text{ann}}_{\mathcal{F}}(m)}{m} = 0$
3. The necessary and sufficient condition for consistency of the ERM principle for any distribution is $\lim_{m\to\infty} \frac{G_{\mathcal{F}}(m)}{m} = 0$.

Finally, we present another capacity concept h, or *VC dimension*, upon which an upper bound of the growth function is further constructed.

Definition *VC dimension* $h \triangleq \max\{m : G_{\mathcal{F}}(m) = m \ln 2\}$, or equivalently, the maximum number of points that can be shattered by $\mathcal{F}$.

It is proved in [23] that for $m > h$, the inequality $G_{\mathcal{F}}(m) \leq h(\ln \frac{m}{h} + 1)$ holds true, leading to the well-known VC bound theorem:

Theorem 2.3.2 *For any function space (countable or uncountable) $\mathcal{F}$, and for any $f \in \mathcal{F}$, the following bound holds true,*

$$\Pr\left\{R_{p(\mathbf{x},y)}(f) \leq R_{emp}(f) + \sqrt{\frac{h(\ln(2m/h) + 1) + \ln(4/\delta)}{m}}\right\} > 1 - \delta \qquad (2.18)$$

There are nice properties about this bound: (1) *distribution-independent* — it holds true regardless of the distribution $p(\mathbf{x}, y)$; (2) *algorithm-independent* — it holds true regardless of a specific choice of f. (3) *data-independent* — it holds true regardless of sample values $\mathcal{D}_m$. The only relevant factor is the VC dimension of $\mathcal{F}$, a measure that reflects the capacity of the function space. The lower the VC dimension, the faster the rate of convergence, and the convergence rate is uniformly bounded for all decision functions in the function space. In other words, the capacity term in Equation (2.16) is essentially $\Phi(\cdot) = \Phi(h(\mathcal{F}), m, \delta)$, which does not depend on specific f or $\mathcal{D}_m$. These independence properties, however, come at the cost that such a bound is generally loose compared with an algorithm-dependent, or a data-dependent bound which we will discuss next.

2.3.2 PAC-Bayesian bounds

The PAC-Bayesian framework [60, 83, 85] combines the advantages of Bayesian methods with PAC learning [86]. This approach inherits the main characteristic of a Bayesian approach by incorporating domain knowledge in the form of a Bayesian prior $\pi(f)$, $f \in \mathcal{F}$. Here we use what we call a "standard prior" $\pi(f)$ to denote the prior in the conventional sense. The standard prior is chosen before seeing any training or test data, which should be distinguished from the "fidelity prior" which will be introduced in Chapter 4. Furthermore, the PAC-Bayesian approach provides a generalization error bound *without assuming the truth of the prior*. The simplest PAC-Bayesian bound is the *Occam's Razor bound*, which is only valid for countable function spaces, *i.e.*, $|\mathcal{F}| < \infty$.

Theorem 2.3.3 (Occam's Razor bound [83, 85]) *For any probability distribution $p(\mathbf{x}, y)$ where the samples are drawn in an i.i.d. fashion, for any prior distribution $\pi(f)$ defined on a countable function space $\mathcal{F}$, and for any f for which $\pi(f) > 0$, the following bound holds true*

$$\Pr\left\{R_{p(\mathbf{x},y)}(f) \leq R_{emp}(f) + \sqrt{\frac{-\ln \pi(f) - \ln \delta}{2m}}\right\} > 1 - \delta. \tag{2.19}$$

A proof of this theorem is provided in Appendix A.1, which utilizes the union bound theorem, thereby limiting the use of the Occam's Razor bound to countable function spaces. The general capacity term in Equation (2.16) is $\Phi(\cdot) = \Phi(\mathcal{F}, f, m, \delta)$, in contrast to the VC bound where $\Phi(\cdot) = \Phi(\mathcal{F}, m, \delta)$. Here $\Phi(\cdot)$ is no longer a capacity measure of the function space but a minimum-description-length-type measure of individual decision functions. Consequently, the convergence rate can be different at different points in the function space.

Similar to the VC bound, Theorem 2.3.3 presents a tradeoff between selecting a model that fits the data well and selecting a model with a high prior probability. This offers some insight into how to choose a prior distribution $\pi(f)$ — we should always assign higher prior probabilities to the functions which are a-priori viewed as likely to fit the data well.

One limitation of the Occam's Razor bound is that it is only valid for countable function spaces. McAllester's PAC-Bayes bound for Gibbs classifiers [60, 83, 85] works for both countable and uncountable functions. A Gibbs classifier is a stochastic classifier given by $f \sim q(f)$, meaning that f is drawn randomly from $q(f)$, where $q(f)$ is a posterior distribution of f given the training data $\mathcal{D}_m$. In this setting, we define the stochastic expected risk as

$$\mathrm{E}_{q(f)}[R_{p(\mathbf{x},y)}(f)] = \mathrm{E}_{q(f)}[\mathrm{E}_{p(\mathbf{x},y)}[Q(f(\mathbf{x}), y)]]$$

and the stochastic empirical risk as

$$\mathrm{E}_{q(f)}[R_{\mathrm{emp}}(f)] = \mathrm{E}_{q(f)}[\frac{1}{m}\sum_{i=1}^{m} Q(f(\mathbf{x}, y)],$$

and McAllester's PAC-Bayes theorem bounds the difference, from one side, between these two quantities.

Theorem 2.3.4 (PAC-Bayes bound for Gibbs classifiers [83]) *For any prior distribution $\pi(f)$ and any posterior distribution $q(f)$, the following holds true,*

$$\Pr\left\{ \mathrm{E}_{q(f)}[R_{p(\mathbf{x},y)}(f)] \leq \mathrm{E}_{q(f)}[R_{emp}(f)] + \sqrt{\frac{D(q(f)||\pi(f)) - \ln\delta + \ln m + 2}{2m-1}} \right\} > 1 - \delta. \tag{2.20}$$

Proofs of this theorem can be obtained in [83, 85, 87]. The general capacity term in Equation (2.16) becomes $\Phi(\cdot) = \Phi(\mathcal{F}, f, \mathcal{D}_m, m, \delta)$. In other words, this is a data-dependent bound, since $q(f)$ is in general estimated from data. However, the theorem holds true for **any** posterior distribution, *i.e.*, it does not constrain how to estimate $q(f)$ from data. In the case of a countable function space, if we commit our choice of decision function to one point f' in the function space, *i.e.* $q(f) = \mathrm{I}(f = f')$, we have

$$\Pr\left\{ R_{p(\mathbf{x},y)}(f') \leq R_{\text{emp}}(f') + \sqrt{\frac{-\ln\pi(f') - \ln\delta + \ln m + 2}{2m-1}} \right\} > 1 - \delta \tag{2.21}$$

which is similar to Theorem 2.3.3. For an uncountable function space, however, the KL-divergence $D(q(f)||\pi(f))$ is not well defined when $q(f) = \delta(f = f')$.

Additionally, we are often interested in a Bayesian predictive classifier, which is a deterministic classifier in the form of $f_{Bayes}(\mathbf{x}) \triangleq \mathrm{E}_{q(f)}[f(\mathbf{x})]$. Note that under this definition the actual decision function is sgn $f_{Bayes}(\mathbf{x})$. Seeger [87] has stated that for $f : \mathcal{X} \to \{\pm 1\}$, the expected error of a Bayesian predictive classifier is bounded by twice the stochastic expected error of a Gibbs classifier under the same posterior distribution. Mathematically,

$$R_{p(\mathbf{x},y)}(f_{Bayes}) \leq 2\,\mathrm{E}_{q(f)}[R_{p(\mathbf{x},y)}(f)], \tag{2.22}$$

we provide our proof of this inequality in Appendix A.2. Therefore, the expected error of a Bayesian predictive classifier is less than twice the bound in Theorem 2.3.5. Furthermore, Schapire [80] provided a margin bound on Bayesian predictive classifiers directly, which was further improved by Langford [88–90] in the PAC-Bayesian setting. The bound in [88] is given by

Theorem 2.3.5 (PAC-Bayes margin bound for Bayesian predictive classifiers [88]) *For any prior distribution $\pi(f)$ and any posterior distribution $q(f)$, and for any margin threshold $\theta > 0$,* *the following holds with probability* $1 - \delta$*:*

$$E_{p(\mathbf{x},y)}[\mathrm{I}(yf_{Bayes}(\mathbf{x}) < 0)] \leq \frac{1}{m}\sum_{i=1}^{m}\mathrm{I}(yf_{Bayes}(\mathbf{x}) < \theta)]+\sqrt{\frac{\theta^{-2}D(q(f)||\pi(f))\ln m + \ln m - \ln\delta}{m}} \tag{2.23}$$

where $f_{Bayes}(\mathbf{x}) = \mathrm{E}_{q(f)}[f(\mathbf{x})]$

2.4 Regularization

The previous section shows an important fact in statistical learning: minimizing only the empirical risk can lead to bad generalization performance when the sample size is small. The error bound theorems, however, have offered us theoretically-justified guidance to avoid this problem. The two types of generalization error bounds introduced in the previous section lead to two different, yet unifiable, approaches to regularization.

2.4.1 Structured risk minimization

According to the VC bound theorem, if two models describe the training data equally well, the model with the smallest VC dimension has better generalization performance. We thus can use the VC dimension as a regularizer in minimizing the empirical risk. Let $\mathcal{F}_1 \subset \mathcal{F}_2 \subset ... \subset \mathcal{F}$ be a sequence of increasingly large function spaces. On one hand, we have

$$h(\mathcal{F}_1) \leq h(\mathcal{F}_2) \leq ... \leq h(\mathcal{F})$$

since if $\mathcal{F}_j$ can shatter m points, then $\mathcal{F}_{j+1}$ must shatter at least m points. On the other hand, it is easy to see that

$$\min_{f\in\mathcal{F}_1} R_{\text{emp}}(f) \geq \min_{f\in\mathcal{F}_2} R_{\text{emp}}(f) \geq ... \geq \min_{f\in\mathcal{F}} R_{\text{emp}}(f)$$

In practice, we would like to minimize a "regularized risk" which take into account both the empirical risk and the capacity of the model. Mathematically, we want to find the index of a function space such that

$$j^* = \underset{j}{\text{argmin}} \left[\left((\min_{f\in\mathcal{F}_j} R_{\text{emp}}(f) \right) + \lambda h(\mathcal{F}_j)\right],$$

where λ is a regularization coefficient. We further use $h(f)$ to denote the VC dimension of the minimum function space that contains f, *i.e.*, $h(f) = h(\mathcal{F}_{j'})$ where $j' = \min\{j : f \in \mathcal{F}_j\}$. Then we have

$$\left(\min_{f\in\mathcal{F}_j} R_{\text{emp}}(f)\right) + \lambda \cdot h(\mathcal{F}_j) \geq \min_{f\in\mathcal{F}_j} \left(R_{\text{emp}}(f) + \lambda \cdot h(f)\right) \tag{2.24}$$

Taking a min operation w.r.t. j on both sides, the inequality turns into an equality which follows,

$$\begin{aligned} \min_j \left(\left(\min_{f\in\mathcal{F}_j} R_{\text{emp}}(f)\right) + \lambda \cdot h(\mathcal{F}_j)\right) &= \min_j \left(\min_{f\in\mathcal{F}_j} \left(R_{\text{emp}}(f) + \lambda \cdot h(f)\right)\right) \\ &= \min_{f\in\mathcal{F}} \left(R_{\text{emp}}(f) + \lambda \cdot h(f)\right) \end{aligned} \tag{2.25}$$

One of the most successful examples that implement the idea of structured risk minimization is the use of large margin hyperplanes [23, 78]. It is relatively easy to derive an upper bound of the VC dimension of hyperplanes, and this upper bound can be readily used as a regularizer in ERM. To see this, we first introduce a theorem regarding the VC dimension of hyperplanes in canonical form with a proof (by Schölkopf [61]) provided in Appendix A.3.

Theorem 2.4.1 [23, 61] *Consider hyperplanes* $\mathbf{w}^T\mathbf{x} = 0$ *in canonical form w.r.t.* $\{\mathbf{x}_i\}_{i=1}^m$*, i.e.,*

$$\min_i |\mathbf{w}^T\mathbf{x}_i| = 1$$

For any $\mathbf{w}$*, the set of decision functions* $f(\mathbf{x}) = \text{sgn}\ \mathbf{w}^T\mathbf{x}$ *satisfying the constraint* $\|\mathbf{w}\| \leq \Lambda$ *has a VC dimension satisfying*

$$h \leq R^2\Lambda^2$$

where $R = \max_i \|\mathbf{x}_i\|$.

Taking into account both the empirical risk and the VC dimension, the optimal hyperplane can be found by solving the following constrained quadratic optimization problem [78],

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^m \xi_i \\ \text{subject to} \quad & \xi_i \geq 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \ \text{ and } \ \xi_i \geq 0. \end{aligned} \tag{2.26}$$

Here ξ_i are slack variables introduced to represent the hinge loss, *i.e.*, $\xi_i = |1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b)|_+$. This problem can be solved using the Lagrangian formulation, and is eventually reduced to a dual optimization problem where $\mathbf{w}$ and b are eliminated:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m \alpha_i\alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle \\ \text{subject to} \quad & 0 \leq \alpha_i \leq \frac{1}{\lambda m} \ \text{ and } \ \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned} \tag{2.27}$$

The solution is given by $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$. The resulting hyperplane is determined by those training samples with nonzero α_i values, known as support vectors (SVs).

2.4.2 Bayesian model selection

An alternative approach to regularization is to take the Bayesian view of the world. We view the model f as a random variable and we specify a prior distribution $\pi(f)$ before seeing the training data. The posterior probability of a decision function is given by

$$p(f|\mathcal{D}_m) = \frac{p(\mathcal{D}_m|f)\pi(f)}{p(\mathcal{D}_m)}, \tag{2.28}$$

In the case of a single, deterministic classifier, we perform maximum a posteriori (MAP) estimation which is essentially a regularized optimization criterion:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} \left(R_{\text{emp}}(f) - \lambda \ln \pi(f) \right) \tag{2.29}$$

Here we substituted $\ln p(\mathcal{D}_m|f)$ with a general risk $R_{\text{emp}}(f)$ which can use any loss function depending on the classifier of interest. Furthermore, we added a regularization coefficient to control how much we trust our prior knowledge. Note that this criterion resembles structured risk minimization except that $-\ln \pi(f)$ replaces $h(f)$ as the regularizer. When the function space is countable, this model selection algorithm is justified by the Occam's Razor bound (Theorem 2.3.3) — in order to guarantee a low expected risk, it is desired to select the model with both a low empirical risk and a high prior probability. Although there is not yet known that there exists an analogy of the Occam's Razor bound for single, deterministic classifiers in uncountable function spaces, we will show in Chapter 5 that it empirically works well to use Equation (2.29) in learning such classifiers.

Similarly, for Gibbs classifiers, McAllester's PAC-Bayesian bound in Theorem 2.3.5 suggests the following training objective:

$$q^*(f) = \operatorname*{argmin}_{q} \left(\mathrm{E}_{q(f)}[R_{\text{emp}}(f)] + \lambda D(q(f)||\pi(f)) \right) \tag{2.30}$$

A similar training objective can be derived for Bayesian predictive classifiers. For example, *maximum entropy discrimination* [91] seeks a posterior distribution $q(f)$ that minimizes $D(q(f)||\pi(f))$ under the constraints that $\mathrm{E}_{q(f)}[\mathrm{E}_{p(\gamma_i)}[(y_i f(\mathbf{x}_i) - \gamma_i)]] \geq 0$, which is theoretically justified by the corresponding error bounds [88, 90].

In both cases, the prior distribution $\pi(f)$ plays a crucial role in the generalization ability of a decision function. Choosing an appropriate prior distribution, just as choosing a right sub-function-space in structured risk minimization, can yield an estimation that attains a low empirical error rate as well as a low capacity term. In fact, structured risk minimization can be viewed as a special case of Bayesian model selection, where the functions $f \in \mathcal{F}_j$ are assigned higher prior probabilities than functions $f \in \mathcal{F}_{j+1} \setminus \mathcal{F}_j$, and where all functions $f \in \mathcal{F}_{j+1} \setminus \mathcal{F}_j$ are assigned equal prior probabilities. Moreover, the choice of a prior distribution has been traditionally a compromise between a realistic assessment of beliefs and choosing a parametric form that simplifies analytical calculations. Certain forms of the prior are preferred due to their mathematical tractability. For example, in the case of generative models, a *conjugate prior* $\pi(f)$ w.r.t. the joint sample distribution $p(\mathbf{x}, y|f)$ is often used, so that the posterior $p(f|\mathbf{x}, y)$ belongs to the same functional family as the prior. In Langford's *PAC-Bayesian margin bound* [92], a Gaussian prior is chosen to simplify mathematical derivations.

2.5 Information Theoretical Background

This section gives information theoretical basics that are necessary to understanding the rest of the work. Information theory was originally presented by Shannon in his paper "*A Mathematical Theory of Communication*", though the discussion of this section is mainly based on Cover's work [84]. Regarding the notation used in this section, we again use $p(x)$ as a short-cut for $p_X(X = x)$, and we in general use scalar representations (*e.g.* x), but bear in mind that all concepts and theorems are valid for vector variables.

The key concept in information theory is *entropy*, which measures the randomness of an event.

Definition The entropy of a random variable $X \in \mathcal{X}$ is defined as $H(X) = -\operatorname{E}_{p(x)}[\log p(x)]$.

If $|\mathcal{X}| < \infty$, $\mathcal{X}$ denotes the alphabet of X; otherwise $\mathcal{X}$ denotes the support set of X and $p(x)$ denotes the probability density function. Note that the term "differential entropy" and the symbol $h(X)$ are often used to represent the entropy in cases $|\mathcal{X}| = \infty$. Here we ignore this distinction and use $H(X)$ for both cases.

The *conditional entropy* measures the randomness of a random variable given another random variable.

Definition The conditional entropy is defined as $H(X|Y) = -\operatorname{E}_{p(x,y)}[\log p(x|y)]$.

It is easy to derive the chain rule $H(X, Y) = H(X|Y) + H(Y)$.

Definition The relative entropy, or the Kullback-Leibler (KL) divergence, between two distributions $p(x)$ and $q(x)$ of a random variable X, is defined as $D(p(x)||q(x)) = \operatorname{E}_{p(x)}[\log \dfrac{p(x)}{q(x)}]$

In the above definition, we use the convention that $0 \log \frac{0}{q} = 0$ and $p \log \frac{p}{0} = \infty$. The relative entropy satisfies many important mathematical properties. For example, $D(p(x)||q(x)) \geq 0$, where the equality holds true iff $p(x) = q(x)$ for all $x \in \mathcal{X}$. Also, $D(p(x)||q(x))$ is a convex function of $p(x)$, as well as of $q(x)$. Similarly, we define the conditional relative entropy as follows.

Definition The conditional relative entropy between two conditional distributions $p(x|y)$ and $q(x|y)$ is defined as $D(p(x|y)||q(x|y)) = \operatorname{E}_{p(x,y)}[\log \dfrac{p(x|y)}{q(x|y)}]$

The chain rule is also applicable to relative entropy: $D(p(x, y)||q(x, y)) = D(p(x|y)||q(x|y)) + D(p(y)||q(y))$.

Next, we study the information theoretical quantities for Gaussian distributions. It is well known that the Gaussian distribution maximizes entropy among all distributions with the first moment equality constraints, and hence is a natural choice of the underlying distribution for many data generating processes. The entropy of a multivariate Gaussian distribution $\mathcal{N}(\mathbf{x}; \mu, \Sigma)$ is $\frac{1}{2} \log(2\pi e)^d |\Sigma|$, where d denotes the dimensionality. The relative entropy of two Gaussian distributions follows

KL-divergence of normal distributions [93]: If $p(\mathbf{x}|\theta_1) = \mathcal{N}(\mathbf{x}; \mu_1, \Sigma_1)$ and $p(\mathbf{x}|\theta_2) = \mathcal{N}(\mathbf{x}; \mu_2, \Sigma_2)$, where $\mathbf{x} \in R^d$, then

$$D(\mathcal{N}(\mathbf{x}; \mu_1, \Sigma_1)||\mathcal{N}(\mathbf{x}; \mu_2, \Sigma_2)) = \frac{1}{2} \ln \left(\frac{|\Sigma_2|}{|\Sigma_1|} \right) + \frac{1}{2} \operatorname{tr} \left(\Sigma_1 \Sigma_2^{-1} + (\mu_2 - \mu_1)^T \Sigma_2^{-1} (\mu_2 - \mu_1) \right) - \frac{d}{2} \tag{2.31}$$

In particular, if $\Sigma_1 = \Sigma_2$, the relatively entropy is a Mahalanobis distance between two Gaussian means.

$$D(\mathcal{N}(\mathbf{x}; \mu_1, \Sigma_1)||\mathcal{N}(\mathbf{x}; \mu_2, \Sigma_2)) = \frac{1}{2}(\mu_2 - \mu_1)^T \Sigma_2^{-1} (\mu_2 - \mu_1) \tag{2.32}$$

If we treat the relative entropy as a function of (μ_2, Σ_2), then its functional form equals that of the negative logarithm of a normal-Wishart density on (μ_2, Σ_2) plus a constant. To see this we write the general form of a normal-Wishart density[2] as

$$\mathcal{W}^{-1}(\mu,\Sigma|\nu,\tau,a,B) \propto |\Sigma|^{-(a-d)/2}\exp\{-\frac{\tau}{2}(\mu-\nu)^T\Sigma^{-1}(\mu-\nu)\}\exp\{-\frac{1}{2}\operatorname{tr}(B\Sigma^{-1})\} \quad (2.33)$$

where (τ,ν,a,B) are hyperparameters. It is easy to see that we have

$$D(\mathcal{N}(\mathbf{x};\mu_1,\Sigma_1)||\mathcal{N}(\mathbf{x};\mu_2,\Sigma_2)) = \mathcal{W}^{-1}(\mu_2,\Sigma_2|\nu,\tau,a,B) + C$$

where $\tau = 1$, $\nu = \mu_1$, $a = d+1$, $B = \Sigma_1$ and $C = -\frac{1}{2}\ln|\Sigma_1| - \frac{d}{2}$. Moreover, if $\Sigma_2 = \Sigma_1$, the relative entropy is proportional to the negative logarithm of a Gaussian distribution.

In fact, the relative entropy can be conveniently calculated if the probability density $p(x|\theta)$, given parameters θ, belongs to the exponential family of distributions, *i.e.*, $p(x|\theta) = \exp\{a(x) + b(\theta) + c(x)d(\theta)\}$. In particular, if $c(x) = x$, the distribution is said to be in canonical form. Many distributions, such as Gaussian, Poisson, binomial, and Gamma distributions, belong to the exponential family in canonical form.

KL-divergence of exponential family distributions: If $p(x|\theta_1) = \exp\{a(x)+b(\theta_1)+c(x)d(\theta_1)\}$, and $p(x|\theta_2) = \exp\{a(x) + b(\theta_2) + c(x)d(\theta_2)\}$, *i.e.* two exponential family distributions with the same parametric form but with different parameters, then

$$D(\mathcal{N}(p(\mathbf{x}|\theta_1)||p(\mathbf{x}|\theta_2)) = \left[b(\theta_1) - b(\theta_2) - \frac{b'(\theta_1)}{d'(\theta_2)}(d(\theta_1) - d(\theta_2))\right] \quad (2.34)$$

Proof

$$\begin{aligned} D(\mathcal{N}(p(\mathbf{x}|\theta_1)||p(\mathbf{x}|\theta_2)) &= \mathrm{E}_{p(x)}[\log\frac{\exp\{a(x)+b(\theta_1)+c(x)d(\theta_1)\}}{\exp\{a(x)+b(\theta_2)+c(x)d(\theta_2)\}}] && (2.35)\\ &= \mathrm{E}_{p(x)}[b(\theta_1) - b(\theta_2) + c(x)(d(\theta_1) - d(\theta_2)] && (2.36)\\ &= b(\theta_1) - b(\theta_2) - \frac{b'(\theta_1)}{d'(\theta_2)}(d(\theta_1) - d(\theta_2)) && (2.37)\end{aligned}$$

The last equality follows since for exponential family distributions $\mathrm{E}[c(x)] = -\frac{b'(\theta)}{d'(\theta)}$. ▮

Another useful result is the KL-divergence of normal-Wishart distributions.

KL-divergence of normal-Wishart distributions: If $p(\mu,\Sigma|\lambda_1) = \mathcal{W}^{-1}(\mu,\Sigma|\mu_1,1,a,\Sigma_1)$ and $p(\mu,\Sigma|\lambda_2) = \mathcal{W}^{-1}(\mu,\Sigma|\mu_2,1,a,\Sigma_2)$

$$D(\mathcal{N}(p(\mu,\Sigma|\lambda_1)||p(\mu,\Sigma|\lambda_2)) = \frac{a}{2}\operatorname{tr}(\Sigma_1\Sigma_2^{-1}) + \frac{a}{2}(\mu_2-\mu_1)^T\Sigma_1^{-1}\mu_2 - \mu_1) \quad (2.38)$$

[2]Since Σ is a covariance matrix rather than a precision matrix, we actually use an inverse-Wishart density

In particular, if $\Sigma_1 = \Sigma_2$, we have

$$D(\mathcal{N}(p(\mu, \Sigma|\lambda_1)||p(\mu, \Sigma|\lambda_2)) = \frac{1}{2}(\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1) \tag{2.39}$$

Finally, we introduce several important inequalities that will be referred to in later chapters.

Theorem 2.5.1 (Jensen's inequality [84]) *If f is a convex function and x is a random variable, then*

$$\mathrm{E}_{p(x)}[f(x)] \geq f(\mathrm{E}_{p(x)}[x])$$

Using the fact that $a \log a$ is strictly convex, and applying Jensen's inequality, we arrive at the following theorem.

Theorem 2.5.2 (Log sum inequality [84]) *For non-negative numbers, a_i and b_i, $i = 1 \dots n$, we have*

$$\sum_{i=1}^{n} a_i \log \frac{a_i}{b_i} \geq \left(\sum_{i=1}^{n} a_i\right) \log \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}$$

with equality iff $\frac{a_i}{b_i} = const.$

Theorem 2.5.3 (Hoeffding's inequality, Hoeffding 1963) *Suppose $X_1 \dots X_n$ are independent random variables with finite first and second moments. Furthermore assume that the X_i are bounded;* i.e. *assume for $1 \leq i \leq n$ that $\Pr\Big(X_i \in [a_i, b_i]\Big) = 1$. Then for the sum of these variables $S = X_1 + \cdots + X_n$ we have the inequality*

$$Pr(S - \mathrm{E}[S] \geq nt) \leq \exp\left(-\frac{2\,n^2\,t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right)$$

holds true for positive values of t.

Chapter 3

REVIEW OF PRACTICAL WORK ON ADAPTATION

There has already been a vast amount of practical work on adaptation in areas of automatic speech recognition (ASR), natural language processing (NLP) and pattern recognition in general. This chapter provides a brief literature review on techniques developed in these areas.

Practical methods of adaptation generally fall into two categories. The first category of methods apply adaptation in the **feature space** to explicitly account for sources of variation. In speech recognition, for example, to discover the sources of speaker variation, it is necessary to understand how speech is produced. In fact, speech is produced as air is expelled from the lungs, pushed through the vibrating vocal folds, and then ”filtered” by the vocal tract (VT). The length of the VT has a substantial effect on the spectrum of the observed acoustic signal, and is a major cause of speaker variability. *Vocal tract length normalization* is a standard method in ASR that explicitly compensates for the difference in VT length [94, 95]. This is often achieved by applying wrapping techniques to the frequency axis. An analogous example in computer vision is illumination variation, which has historically been a challenging problem in object recognition [96]. The difference in illumination between the training set and the test images can be accounted for by extracting somewhat illumination-invariant representations, such as the *quotient image* [97] and the *spherical harmonic subspace* [98].

Although feature-space adaptation is in general very powerful as it directly addresses the essence of the problem, the design of which strongly relies on domain knowledge. In contrast, a second category of methods apply adaptation in the **model space** without assuming any knowledge of the cause of mismatch. Such methods are conceptually simple to develop, and are potentially applicable to different types of adaptation problems. Essentially, model-space adaptation retrains or transform unadapted model parameters to match the characteristics of the adaptation data, while applying certain regularization to avoid overfitting. This category includes a large number of adaptation techniques in areas of ASR, NLP and pattern recognition, as well as the work that will be developed in this dissertation. The following text reviews some major model-space adaptation methods, and overviews our proposed approach.

3.1 In Automatic Speech Recognition

The idea of adaptation has been largely investigated in ASR, especially for systems using continuous HMMs where the observation distributions are represented by Gaussian mixture models (GMMs). A major difficulty in speech recognition is speaker variability due to different vocal tract lengths, accents and idiosyncrasies (as well as mismatch in channel and noise conditions). Speaker adaptation, which enables the unadapted model to capture the characteristics of the target speaker using a small amount of adaptation data, has become one of the crucial techniques that any state-of-the-art ASR system cannot do without.

Maximum likelihood linear regression (MLLR) is one popular framework for adapting Gaussian mixture HMMs [99, 100], where clusters of model parameters are transformed through shared affine functions. These transformations shift the means and alter the covariance matrices of the Gaussians so that each HMM state is more likely to generate the adaptation data. During recognition, a speaker-dependent transformation is applied to the unadapted model to generate a speaker-dependent model. Formally, we use $\mathbf{x}_i$ to denote the input feature vector of the i^{th} adaptation sample, and use y_i and k_i to denote its hidden Gaussian mixture ID (or equivalently state ID) and hidden component ID respectively. The Gaussian parameters are represented by $f = (\mu_{y,k}, \Sigma_{y,k})$, where $\mu_{y,k}$ denotes the mean of the k^{th} component of the y^{th} Gaussian mixture; and similarly for $\Sigma_{y,k}$. Furthermore, we use a superscript tr to indicate unadapted model parameters, and we temporarily assume that all Gaussian components share the same transformation. Mathematically, the adapted mean is given by

$$\hat{\mu}_{y,k} = A\mu^{tr}_{y,k} + b = W\xi^{tr}_{y,k} \tag{3.1}$$

where $\xi^{tr}_{y,k} = [\mu^{tr}_{k,1}\, \mu^{tr}_{k,2} \ldots \mu^{tr}_{k,d}\, 1]^T$ is the extended unadapted mean; and the adapted covariance matrix is given by

$$\hat{\Sigma}_{y,k} = H\Sigma^{tr}_{y,k}H^T \tag{3.2}$$

The goal is to find W and H that maximize the *incomplete likelihood* of the adaptation data, *i.e.*,

$$\max_{W,H} \ln p(\mathbf{x}_{1:m}|f^{tr}, W, H) \tag{3.3}$$

The optimal parameters are found using an EM approach [63] which iteratively maximizes a lower bound of the incomplete likelihood in Equation (3.3)

$$\max_{W,H} \sum_y \sum_k \sum_{i=1}^m L_{y,k}(i) \ln \mathcal{N}(\mathbf{x}_i; \hat{\mu}_{y,k}, \hat{\Sigma}_{y,k}) \tag{3.4}$$

where $L_{y,k}(i) \triangleq p(y_i = y, k_i = k|\mathbf{x}_{1:m}, f^g)$. It is worth noting that covariance adaptation is generally less effective than mean adaptation and is less commonly used [101]. Moreover, MLLR adaptation can be applied in a flexible manner. For example, when the adaptation data is extremely limited, we can apply a global transformation (W, H) to all Gaussian means and covariance matrices; and when the adaptation is abundant, we use different transformations for different clusters of GMMs. A standard approach is to use a *regression class tree* [102], which clusters model parameters hierarchically and controls the number of transformations based on the amount of adaptation data available.

A second important model-space adaptation technique is Bayesian *maximum a posteriori* (MAP) [103–105], which involves the use of prior knowledge about model parameters. According to the Bayes rule, maximizing the posterior probability $p(f|\mathbf{x}_{1:m})$ is equivalent to

$$\max_f \left(\ln p(\mathbf{x}_{1:m}|f) + \ln p(f) \right) \tag{3.5}$$

where $p(\mathbf{x}_{1:m}|f)$ is the incomplete likelihood and $p(f)$ is a prior distribution of f. There are three key problems regarding MAP estimation [104]: (1) how to define the functional form of the prior; (2) how to estimate the hyper-parameters of the prior; and (3) how to estimate model parameters given (1) and (2). The first problem is typically solved by using the conjugate prior of the complete likelihood so that the posterior belongs to the same functional family as the prior. In our case the complete likelihood is a Gaussian, and its conjugate prior is given by Equation (2.33) which is repeated below for convenience.

$$\begin{aligned} & \mathcal{W}^{-1}(\mu_{y,k}, \Sigma_{y,k}|\nu_{y,k}, \tau_{y,k}, a_{y,k}, B_{y,k}) \\ \propto \quad & |\Sigma|^{-(a_{y,k}-d)/2} \exp\{-\tfrac{\tau_{y,k}}{2}(\mu_{y,k} - \nu_{y,k})^T \Sigma^{-1}(\mu - \nu)\} \exp\{-\tfrac{1}{2} \operatorname{tr}(B_{y,k}\Sigma^{-1})\} \end{aligned} \tag{3.6}$$

Given such a prior, it is not difficult to estimate the model parameters using the EM algorithm, as described in [104]. The updated mean and covariance matrix at each EM iteration are given by

$$\hat{\mu}_{y,k} = \frac{\tau_{y,k}\nu_{y,k} + \sum_{i=1}^{m} L_{y,k}(i)\mathbf{x}_i}{\tau_{y,k} + \sum_{i=1}^{m} L_{y,k}(i)} \tag{3.7}$$

$$\hat{\Sigma}_{y,k} = \frac{B_{y,k} + \sum_{i=1}^{m} L_{y,k}(i)(\mathbf{x}_i - \hat{\mu}_{y,k})(\mathbf{x}_i - \hat{\mu}_{y,k})^T + \tau_{y,k}(\nu_{y,k} - \hat{\mu}_{y,k})(\nu_{y,k} - \hat{\mu}_{y,k})^T}{a_{y,k} - d + \sum_{i=1}^{m} L_{y,k}(i)} \tag{3.8}$$

It can be seen that the updated parameters are determined by two components, namely the hyper-parameters and the adaptation data, and that $\tau_{y,k}$ serves as a weight associated with the k^{th} component of the y^{th} Gaussian mixture. Now the remaining question is how to estimate the hyper-parameters $\nu_{y,k}$, $\tau_{y,k}$, $a_{y,k}$ and $B_{y,k}$. There are different way of doing this, depending on different types of applications [103]. Since we are interested in speaker adaptation, these hyper-parameters can be derived from the unadapted model parameters. Specifically, [103] proposed a set of estimates, which are re-written in our notation as

$$a_{y,k} = \frac{\tau_{y,k} + 1}{2} \tag{3.9}$$

$$\nu_{y,k} = \mu_{y,k}^{tr} \tag{3.10}$$

$$B_{y,k} = \frac{\tau_{y,k}}{2}\Sigma_{y,k}^{tr} \tag{3.11}$$

The hyper-parameter $\tau_{y,k}$ are empirically estimated from data [103]. In practice, to increase robustness, the values of $\tau_{y,k}$ are often constrained to be identical across all y and k (*i.e.* all Gaussians in the system) [103]. Notice that a fundamental property of MAP adaptation is its asymptotical convergence to maximum likelihood estimation when the amount of adaptation data increases. However, without any structural assumption, MAP adaptation only updates parameters of those Gaussians that have observations and thus converges slowly in a system with many Gaussians.

There are various techniques to combine the structural information captured by linear regression with the prior knowledge utilized by Bayesics. *Maximum a posteriori linear regression* (MAPLR) and its variations [106–108] improve over MLLR by assuming a prior distribution on affine transformation parameters. Mathematically,

$$\max_{W,H} \left(\ln p(\mathbf{x}_{1:m}|f^{tr}, W, H) + \ln p(W, H) \right) \tag{3.12}$$

Again, conjugate priors $p(W)$ and $p(H)$ (assumed independent) are typically used, which are in this case Wishart distributions.

Additionally, instead of using point estimates of the transformation parameters in prediction, researchers have been applying full Bayesian inference in some situations to enhance the robustness of adaptation [109–111]. Specifically, let T represent some transformation applied to the unadapted model f^{tr}, *e.g.*, $T = (W, H)$ in the case of MLLR. At test time, there are two ways of computing the posterior probability of a state sequence given an input sequence. One is to compute $p(y_{1:m}|\mathbf{x}_{1:m}, f^{tr}, \hat{T})$ where $\hat{T}$ is a point estimate of the transformation parameters learned using methods like MLLR or MAPLR. The other way is to apply full Bayesian inference which marginalizes out the transform parameters as follows,

$$p(y_{1:m}|\mathbf{x}_{1:m}, f^{tr}) = \int_T p(y_{1:m}|\mathbf{x}_{1:m}, f^{tr}, T)q(T) \tag{3.13}$$

where $q(T)$ is a posterior distribution estimated from the adaptation data. This latter approach is in general more robust to estimation and modeling errors when only a limited amount of adaptation data is available [109, 110].

Another important family of adaptation techniques are conducted in the framework of *speaker adaptive training* (SAT) [112]. This framework utilizes speaker adaptation techniques, such as MLLR or MAPLR, during training to explicitly address speaker-induced variations. Specifically, SAT jointly estimates a compact unadapted model f_c^{tr} and a set of speaker-dependent transformations $T^1, T^2, \ldots, T^K$ (applied to f_c^{tr}) that maximize the likelihood of all speaker-specific training sets $\mathbf{x}^k_{1:n}$, *i.e.*

$$\max_{f_c^{tr}, T^1, .., T^k} \sum_{k=1}^{K} \ln p(\mathbf{x}^k_{1:n}|f_c^{tr}, T^k) \tag{3.14}$$

Since speaker variability has been explicitly accounted for by the transformations in training, the resulting f_c^{tr} only needs to address intrinsic phonetic variability and is hence more compact than a conventional speaker-independent model. During recognition, f_c^{tr} is treated as the unadapted model, and a transformation for the target speaker is estimated using the adaptation data. SAT is in fact a practical application of multi-task learning we introduced in Chapter 1.

There are a few extensions to this framework based on the notion of ”speaker clusters” [113, 114]. For example, [114] proposed *cluster adaptive training* where all Gaussian components in the system are partitioned into R Gaussian classes, and all training speakers are partitioned into P speaker clusters. It is assumed that a speaker-dependent model (either in adaptive training or in recognition) is a linear combination of cluster-conditional models, and that all Gaussian components in the same Gaussian class share the same set of weights. Specifically, for ”model-based clusters” [114], the adapted mean of a Gaussian component (indexed as (y, k) for consistency with earlier equations) is given by

$$\hat{\mu}_{y,k} = \sum_{p=1}^{P} \lambda_p^{r(y,k)} \cdot \mu_{y,k}^{p} \tag{3.15}$$

where $\mu_{y,k}^{p}$ is the mean from the p^{th} speaker cluster, λ_p^r is its associated weight, and $r(y, k) \in \{1..R\}$ indicates which Gaussian class the Gaussian component (y, k) belongs to. Similarly, for ”transform-based clusters” [114], the adapted mean of a Gaussian component is given by

$$\hat{\mu}_{y,k} = (\sum_{p=1}^{P} \lambda_p^{r(y,k)} \cdot W^p)\xi_{y,k}^{tr} \tag{3.16}$$

where ξ_c^{tr} is the unadapted mean, and W^p is the affine transformation from the p^{th} speaker cluster and $\lambda_p^{r(\mathbf{x},y)}$ is its associated weight. In both cases, we want to estimate the class-conditional parameters, *i.e.* $\mu_{y,k}^{p}$ or W^p, and the weights λ_p^r that maximize the likelihood function as expressed by Equation (3.4). The only difference is that the parameters $\hat{\mu}$ therein is replaced by Equation (3.15) or (3.16). There is no simple solution to jointly estimating the cluster-dependent parameters and the weights; an iterative approach is typically used where one set of parameters are updated while the other set of parameters are fixed [114].

In a similar spirit, *eigenvoice* [115] also constrains a speaker-dependent model to be a linear combination of a number of basis models. The difference is that it creates a speaker-dependent supervecctor by concatenating the mean vectors of all HMM Gaussian components, then performs principle component analysis on the supervectors of all training speakers, producing the so-called eigenvoices. During recognition, a new speaker’s supervector is a linear combination of eigenvoices where the weights are estimated to maximize the likelihood of the adaptation data. Eigen-analysis also has been applied to a transformed feature space using the ”kernel trick” [116], or applied to a space of affine transformations which leads to eigen-space MLLR [117, 118].

3.2 In Natural Language Processing

In NLP applications, the domain adaptation problem arises very frequently as great human efforts have been spent annotating text resources for morphological, syntactic and semantic information

[119]. This section reviews a few model-space algorithms for domain adaptation.

First, a simple approach to n-gram language model adaptation is Bayesian MAP adaptation [120], as shown in Equation (3.5). In n-gram language models, the underlying goal is to produce conditional probability representations of the form $p(w_t = j|h_t)$, where $w_t \in \{1..j\}$ is the word at position t and h_t is the history. In a trigram model, for example, $h_t = (w_{t-1}, w_{t-2})$. For each value of $h_t = h$, we define $\omega_{j|h} \overset{\Delta}{=} p(w_t = j|h_t = h)$ for simplicity, and obviously $\sum_{j=1}^{N} \omega_{j|h} = 1$. The conjugate prior for this distribution is the Dirichlet distribution [121]

$$g(\omega_{1|h}, \omega_{2|h}, \dots \omega_{N|h}|\alpha_1, \alpha_2, \dots, \alpha_N) \propto \prod_{j=1}^{N} \omega_{j|h}^{\alpha_j - 1}; \tag{3.17}$$

and the adapted n-gram probabilities are computed as

$$\hat{\omega}_j = \frac{(\alpha_j - 1) + c(h,j)}{\sum_{j=1}^{N}(\alpha_j - 1) + \sum_{j=1}^{N} c(h,j)} \tag{3.18}$$

where $c(h,j)$ is the expected count of the ngram $(h_t = h, w_t = j)$ in the adaptation data. The unadapted model parameters can be utilized in choosing the hyperparameters of the Dirichlet distribution, as was discussed in [120].

The use of a prior distribution on model parameters has also been applied to the adaptation of conditional models. Here again we use $\mathbf{x}_i$ to denote an input feature vector and use y_i to denote its label. In NLP applications, $\mathbf{x}_i$ is usually word-level features such as word IDs and context word IDs, and y_i can be a part-of-speech tag, a capitalization indicator, or a name entity indicator depending on specific applications. The goal is to adapt the conditional model parameters to maximize $p(f|\mathbf{x}_{1:m}, y_{1:m}) \propto p(y_{1:m}|\mathbf{x}_{1:m}, f)p(f)$ w.r.t. the adaptation data. For example, [122] presented an algorithm for adapting conditional maximum entropy models for automatic capitalization. This algorithm incorporates the information from the unadapted model by using a Gaussian prior on the feature weights $w_j, j = 1..N$, resulting in the following adaption objective,

$$\max_{w_{1:N}} \left(\ln p(y_{1:m}|\mathbf{x}_{1:m}, w_{1:N}) - \sum_j \frac{(w_j - w_j^{tr})^2}{\sigma_j^2} \right) \tag{3.19}$$

where $w_{1:N}$ are the unadapted feature weights.

Furthermore, analogous to speaker clustering in speech recognition, there are mixture model based approaches to both n-gram model adaptation [123] and conditional maximum entropy model adaptation [119]. In a recent work [119], three distributions are modeled in parallel, an in-domain distribution $p^{(i)}(\mathbf{x}, y)$, an out-of-domain distribution $p^{(o)}(\mathbf{x}, y)$, and a general distribution $p^{(g)}(\mathbf{x}, y)$. The training distribution is assumed to be a mixture of $p^{(i)}$ and $p^{(g)}$; and the target distribution is assumed to be a mixture of $p^{(o)}$ and $p^{(g)}$. In both cases, the mixture component identity is modeled using a hidden variable z. Letting superscript tr and ad denote training and adaptation samples respectively, the learning objective is given by

$$\begin{aligned} & \max_f \left(\ln p(y_{1:n}^{tr} | \mathbf{x}_{1:n}^{tr}, f) + \ln p(y_{1:m}^{ad} | \mathbf{x}_{1:m}^{ad}, f) \right) \\ = & \max_f \left(\sum_{i=1}^{n} \ln \sum_{z \in \{i,g\}} p(y_i^{tr}, z_i^{tr} = z | \mathbf{x}_i^{tr}, f) + \sum_{i=1}^{m} \ln \sum_{z \in \{o,g\}} p(y_i^{ad}, z_i^{ad} = z | \mathbf{x}_i^{ad}, f) \right); \end{aligned} \tag{3.20}$$

and the optimal parameters are estimated using the conditional EM algorithm [124].

It is worth mentioning that feature-space adaptation techniques, which capture intrinsic structures at the syntax and semantic level, have drawn increasing attention in parsing, part-of-speech tagging and other NLP applications [125, 126], though they are beyond the scope of this work.

3.3 In Pattern Recognition

So far, we have seen work on adaptation of Gaussian mixture HMMs (for acoustic modeling), of n-gram models (for language modeling) and of conditional MaxEnt models (for POS-tagging or other NLP applications). There are many other statistical models that are actively used in various pattern classification tasks, such as support vector machines (SVMs) and multi-layer perceptrons (MLPS). Here we overview adaptation techniques developed for these classifiers, and more descriptions will be given in Chapter 5 along with the discussion of our work.

The adaptation of MLPs has been tackled from a meta-learning perspective. Both [31] and [26] proposed to construct MLPs whose input-to-hidden layer is shared by multiple related tasks. This layer represents an ”internal representation” which, once learned, is fixed for future learning. In this regard, MLP adaptation amounts to training the hidden-to-output layer for the target task while keeping the input-to-hidden layer ”representation”. This approach has been explicitly applied to adaptation tasks in [127]. Another popular approach to MLP adaptation is adding augmentative layers whose parameters are estimated from the adaptation data. The linear input network approach [128, 129], for example, applies a linear transformation to the input space, where the transformation parameters are learned using the adaptation data.

SVM adaptation, on the other hand, is typically done by combining the support vectors from the unadapted model with a subset of the adaptation data, and then retraining an SVM using the combined data [130–133]. Specifically, [130] combined the old SVs with the adaptation samples mis-classified by the unadapted classifier, while [131] chose to use the correctly-classified samples instead. In [132] and [133], the old SVs and the adaptation data were weighted differently in the optimization objective. Chapter 5 will give a detailed review of these algorithms and discuss how they relate to our proposed approaches.

In summary, adaptation algorithms have been developed for a variety of statistical models, including Gaussian mixture models (GMMs), hidden Markov models (HMMs), support vector machines (SVMs), multi-layer perceptrons (MLPs) and conditional maximum entropy (MaxEnt) models, each of which has been approached differently. While these algorithms have demonstrated empirically the effectiveness of adaptation, it is curious to ask whether there is a principled approach that unifies these different treatments. Moreover, a more fundamental question would be whether we can relate the adaptation error bound to the divergence between training and target distributions. We seek answers to these questions in the next chapter.

Chapter 4

A FIDELITY PRIOR FOR CLASSIFIER ADAPTATION

Recall that $(\mathbf{x}, y) \in \mathcal{X} \times \{\pm 1\}$ is a pair of (input, label) variables with a joint distribution $p(\mathbf{x}, y)$. Inductive learning aims to learn a decision function $f \in \mathcal{F}$ that not only correctly classifies observed samples drawn from $p(\mathbf{x}, y)$, but also generalizes to unseen samples drawn from the *same* distribution. In other words, we desire to learn an f that minimizes the true risk $R_{p(\mathbf{x},y)}(f)$ under certain loss function $Q(\cdot)$. In practice, this is often approached by minimizing the empirical risk $R_{\text{emp}}(f)$ on a training set, while utilizing certain regularization strategy to guarantee good generalization performance, as was discussed in Chapter 2.

The target (or test-time) distribution, however, is often *different* from the training distribution. Sometimes, the difference only resides in the input distribution $p(\mathbf{x})$, while the conditional relation $p(y|\mathbf{x})$ remains the same. In several learning paradigms, this type of difference has been partially accounted for by explicitly taking into account the test input distribution [19,134]. A learning setting that has not received as much theoretical attention is that of ”adaptive learning”, which studies a more general case where *both* $p(\mathbf{x})$ and $p(y|\mathbf{x})$ at test time vary from their training counterparts. This is in fact a common assumption in ASR, where the training set consists of enormous speakers but the application only sees one speaker at a time. Another distinctive assumption of adaptive learning is that while there may be essentially an unlimited amount of labeled training distribution data, only a small amount of labeled adaptation data drawn from the target distribution is available.

To formally define the adaptive learning paradigm, we let $p^{tr}(\mathbf{x}, y)$ and $p^{ad}(\mathbf{x}, y)$ denote the training and target distributions respectively, and we assume that two sources of information are given in a priori:

1. An “unadapted classifier” $f^{tr} \in \text{argmin}_{f \in \mathcal{F}} R_{p^{tr}}(f)$, which is trained using a sufficient amount of training data (but this data is in general *not* preserved for adaptation);

2. “Adaptation data” $\mathcal{D}_m^{ad} = \{(\mathbf{x}_i, y_i) | (\mathbf{x}_i, y_i) \sim p^{ad}(\mathbf{x}, y)\}_{i=1}^{m}$.

The goal of adaptation is to produce an "adapted classifier" $\hat{f}$ (a point estimate) that is as close as possible to our "desired classifier",

$$f^{ad} \in \operatorname{argmin}_{f \in \mathcal{F}} R_{p^{ad}}(f)$$

In this setting, adaptation is supervised, as both training and adaptation data are labeled; it is also inductive, as the adapted classifier is desired to generalize to unseen data drawn from $p^{ad}(\mathbf{x}, y)$. But adaptation can be unsupervised or transductive with modified assumptions. There are two extreme strategies for learning $\hat{f}$. First, we can train a classifier that minimizes the empirical risk $R_{\text{emp}}(f)$ on $(\mathbf{x}_i, y_i) \in \mathcal{D}_m^{ad}$, but this might cause overfitting with small m; even if we apply certain forms of regularization to reduce the variance, the estimate $\hat{f}$ might have a high bias if the regularizer makes "wrong" preferences. At the other extreme, we can simply let $\hat{f} = f^{tr}$, but this might yield a high empirical risk on $\mathcal{D}_m^{ad}$ (again due to high bias), especially when $p^{ad}(\mathbf{x}, y)$ significantly differs from $p^{tr}(\mathbf{x}, y)$. This work seeks a strategy between these two extremes in which one would hope to achieve better performance.

As was reviewed in Chapter 3, there has been a vast amount of practical work on adaptation in the areas of ASR, NLP and pattern recognition, involving a variety of generative and discriminative classifiers. It is interesting to ask whether there is a principled and unified approach to adaptation that is applicable to different types of classifiers. Moreover, a more fundamental question would be whether we can relate the adaptation sample complexity to the divergence between training and target distributions. This chapter makes an attempt to answer these questions. We utilize the concept of "accuracy-regularization", where we seek a classifier that, on one hand, attains low empirical risk on adaptation data, and on the other hand, has good generalization ability as measured by a regularizer. Specifically, we use a Bayesian "fidelity prior" as the regularizer, which leads to principled adaptation strategies for a variety of classifiers. Furthermore, in the PAC-Bayesian setting, this prior relates the adaptation error bound (or sample complexity bound) to the divergence between training and target distributions. The rest of the chapter is organized as follows. Section 4.1 introduces our proposed fidelity prior; Section 4.2 and Section 4.3 discuss its instantiations for generative and discriminative classifiers respectively; and Section 4.4 provides PAC-Bayesian error bound analysis. Throughout this work, we use the symbol 'tr' to indicate parameters of the unadapted classifier and use 'ad' to denote parameters of our desired classifier for the target distribution.

4.1 A Bayesian Fidelity Prior

We approach the adaptation problem from a Bayesian perspective by assuming that f itself is a random variable with a "standard" prior distribution $\pi(f)$ (which is chosen before seeing any training or test data, usually based on domain knowledge), where $\pi(f)$ is defined on a function space $\mathcal{F}$ (either countable of uncountable). In adaptation, we utilize the concept of "accuracy-regularization", where we minimize the empirical risk on the adaptation data while maximizing a fidelity prior [1] $p_{\text{fid}}(f)$ (which will be defined shortly) as follows,

$$\min_{f \in \mathcal{F}} \left[R_{\text{emp}}(f) - \lambda \ln p_{\text{fid}}(f) \right]. \tag{4.1}$$

Note that both $\pi(f)$ and $p_{\text{fid}}(f)$ are Bayesian priors; the difference is that the former is chosen *before* training the unadapted classifier, whereas the latter is chosen *after* the unadapted classifier is obtained. Specifically, the fidelity prior is defined as

$$\ln p_{\text{fid}}(f) \triangleq \mathrm{E}_{p^{tr}(\mathbf{x},y)}[\ln p(f|\mathbf{x},y)] + \gamma \tag{4.2}$$

In this definition, $p^{tr}(\mathbf{x},y)$ again is the training distribution, $p(f|\mathbf{x},y)$ is the posterior probability of a classifier given a sample, and γ is a normalization constant such that $p_{\text{fid}}(f)$ sums to unity. This prior essentially can be viewed as an approximate posterior of a classifier given a training distribution. This resembles the idea of the hierarchical Bayes approach, *e.g.* [33]. The key difference is that the fidelity prior is an *expected log* posterior of a classifier given a sample, while in [33] the prior was the posterior of classifier given a specific sample set. The reason we choose such a prior is that, as will be seen shortly, $p_{\text{fid}}(f)$ incorporates information from both the standard prior $\pi(f)$ and the unadapted classifier f^{tr}, and that it assigns higher probabilities to classifiers "closer to" f^{tr}. More importantly, the choice of this prior analytically relates $p_{\text{fid}}(f^{ad})$ (the prior probability of the *desired* classifier), and hence the generalization error bound at f^{ad}, to the divergence between training and target distributions. Our adaptation objective in Equation (4.1), therefore, becomes a tradeoff between the goodness of data fitting and the fidelity to the unadapted classifier. This fidelity prior leads to a unified adaptation strategy applicable to a variety of classifiers. Next, we discuss its instantiations for generative and discriminative classifiers respectively.

[1] We called it a "divergence prior" in [48]

4.2 Generative Classifiers

We first explore the instantiation of $p_{\text{fid}}(f)$ for classifiers using generative models. In such a case, the function space $\mathcal{F}$ consists of generative models f that describe the sample distribution $p(\mathbf{x}, y|f)$ (here we slightly abuse notation by letting f denote a generative model instead of a decision function). The classification decision is made via

$$\operatorname*{argmax}_{y\in\mathcal{Y}} \ \ln p(\mathbf{x}, y|f)$$

If we use the joint likelihood loss (Equation (2.7)), then the unadapted model

$$f^{tr} \in \operatorname*{argmin}_{f\in\mathcal{F}} R_{p^{tr}(\mathbf{x},y)}(f)$$

is the *true* model generating the training distribution, *i.e.*, $p(\mathbf{x}, y|f^{tr}) = p^{tr}(\mathbf{x}, y)$. Similarly, we have $p(\mathbf{x}, y|f^{ad}) = p^{ad}(\mathbf{x}, y)$. Note that by doing this, we implicitly assume that our function space $\mathcal{F}$ contains the true generative models in both cases, which is standard in PAC learning [86]. Furthermore, applying Bayes rule, the posterior probability in Equation (4.2) can be expressed as

$$p(f|\mathbf{x}, y) = \frac{p(\mathbf{x}, y|f)\pi(f)}{p(\mathbf{x}, y)} = \frac{p(\mathbf{x}, y|f)\pi(f)}{\int p(\mathbf{x}, y|f)\pi(f)\, df} \tag{4.3}$$

where $\pi(f)$ is again the standard prior chosen before seeing the training data. Plugging Equation (4.3) into (4.2) leads to the following theorem,

Theorem 4.2.1 *For generative classifiers, the fidelity prior defined in Equation* (4.2) *satisfies*

$$-\ln p_{\mathit{fid}}(f) = D(f(\mathbf{x}, y|f^{tr})||p(\mathbf{x}, y|f)) - \ln \pi(f) - \beta \tag{4.4}$$

where $\beta > 0$ *is a constant.*

Proof

$$\begin{aligned} -\ln p_{\text{fid}}(f) &= -\int p(\mathbf{x}, y|f^{tr}) \ln p(f|\mathbf{x}, y)\, d\mathbf{x}\, dy - \gamma \\ &= -\int p(\mathbf{x}, y|f^{tr}) \ln[\frac{p(\mathbf{x}, y|f)\pi(f)}{p(\mathbf{x}, y|f^{tr})} \cdot \frac{p(\mathbf{x}, y|f^{tr})}{p(\mathbf{x}, y)}]\, d\mathbf{x}\, dy - \gamma \\ &= D(p(\mathbf{x}, y|f^{tr})||p(\mathbf{x}, y|f)) - \ln \pi(f) - D(p(\mathbf{x}, y|f^{tr})||p(\mathbf{x}, y)) - \gamma \end{aligned} \tag{4.5}$$

Letting $\beta = D(p(\mathbf{x}, y|f^{tr})||p(\mathbf{x}, y)) + \gamma$, we have

$$\begin{aligned} 1 = \int p_{\text{fid}}(f)\, df &= \int_{\mathcal{F}} \exp\{-D(p(\mathbf{x}, y|f^{tr})||p(\mathbf{x}, y|f)) + \ln \pi(f) + \beta\}\, df \\ &< \int_{\mathcal{F}} \exp\{\ln \pi(f) + \beta\}\, df = e^{\beta} \end{aligned} \tag{4.6}$$

The inequality follows that $D(p(\mathbf{x}, y|f^{tr})||p(\mathbf{x}, y)) \geq 0$. Furthermore, since $D(p(\mathbf{x}, y|f^{tr})||p(\mathbf{x}, y)) = 0$ is only achieved at $f = f^{tr}$ in the integral, the inequality $\beta > 0$ is strict.

This fidelity prior is essentially determined by the KL-divergence between the sample distribution generated by the unadapted model and that generated from the model of interest, and it favors those models similar to the unadapted model. In particular, we inspect the prior probability of our desired model, *i.e.*, $\ln p_{\text{fid}}(f^{ad}) = -D(p^{tr}||p^{ad}) + \ln \pi(f^{ad}) + \beta$, from which we can draw some intuitive insights about why using the fidelity prior would help. As implied in the above equation, if $D(p^{tr}||p^{ad}) < \beta$, we have $p_{\text{fid}}(f^{ad}) > \pi(f^{ad})$, and thus we are more likely to learn the desired model using the fidelity prior than using the standard prior. Since $\beta > 0$, for any f^{tr}, there must exist distributions p^{ad} for which the above statement is true. In Section 4.4, we will elaborate on this implication from an error bound perspective.

Consequently, our adaptation objective for generative classifiers becomes

$$\min_{f} R_{\text{emp}}(f) + \lambda D(p(\mathbf{x}, y|f^{tr})||p(\mathbf{x}, y|f)) - \lambda \pi(f) \tag{4.7}$$

This is similar to the objective in [93, 135] for iterative training of GMM and HMM parameters. The key difference is that [93, 93] takes a frequentist approach where the KL-divergence is treated as an "entropic distance", while we take a Bayesian approach where the KL-divergence is derived from a prior distribution of the classifier. When $\pi(f)$ is uniform[2], this objective asks to minimize the empirical risk as well as the KL-divergence between the joint distributions.

In the following text, we discuss instantiations of the fidelity prior and the resulting adaptation objectives for specific sample distributions. We assume that $p(\mathbf{x}, y|f^{tr})$ and $p(\mathbf{x}, y|f^{ad})$ belong to the same distribution family (*e.g.* Gaussian distributions) but with different parameters. In learning the adapted model, we keep the parametric form of the model and search for the optimal parameters. Hence f here is represented by model parameters.

[2]Although improper on unbounded support, a uniform prior does not cause problems in a Bayesian analysis as long as the posterior corresponding to this prior is integrable.

4.2.1 Gaussian models

The KL-divergence, and hence the fidelity prior, can be expressed analytically if the class-conditional distributions are Gaussians, *i.e.*, $p(\mathbf{x}|y, f^{tr}) \sim \mathcal{N}(\mathbf{x}; \mu_y^{tr}, \Sigma_y^{tr})$ and $p(\mathbf{x}|y, f) \sim \mathcal{N}(\mathbf{x}; \mu_y, \Sigma_y)$. We also define the class prior probabilities $p(y|f^{tr}) = \omega_y^{tr}$ and $p(y|f) = \omega_y$. Thus f is represented by $(\omega_y, \mu_y, \Sigma_y)$.

Corollary 4.2.2 *For class-conditional Gaussian models, the fidelity prior $p_{\textit{fid}}(f)$ defined in Equation* (4.4) *satisfies*

$$\begin{aligned} -\ln p_{\textit{fid}}(f) = \sum_y \frac{\omega_y^{tr}}{2} \left[\ln\left(\frac{|\Sigma_y|}{|\Sigma_y^{tr}|}\right) + \mathrm{tr}\left(\Sigma_y^{tr}\Sigma_y^{-1} + (\mu_y - \mu_y^{tr})^T \Sigma_y^{-1} (\mu_y - \mu_y^{tr})\right) - d \right] \\ + \sum_y \omega_y^{tr} \ln \frac{\omega_y^{tr}}{\omega_y} - \ln \pi(f) - \beta \end{aligned} \tag{4.8}$$

where $\beta > 0$. In particular, if $\omega_y = \omega_y^{tr}$, $\Sigma_y = \Sigma_y^{tr}$, we have

$$-\ln p_{\textit{fid}}(f) = \sum_y \frac{1}{2} \omega_y^{tr} (\mu_y - \mu_y^{tr})^T {\Sigma_y^{tr}}^{-1} (\mu_y - \mu_y^{tr}) - \ln \pi(f) - \beta \tag{4.9}$$

Proof Applying Theorem 4.2.1, we have

$$\begin{aligned} -\ln p_{\text{fid}}(f) &= D(p(\mathbf{x}, y|f^{tr})||p(\mathbf{x}, y|f)) - \ln \pi(f) - \beta \\ &= D(p(\mathbf{x}|y, f^{tr})||p(\mathbf{x}|y, f)) + D(p(y|f^{tr})||p(y|f)) - \ln \pi(f) - \beta \\ &= \sum_y \omega_y^{tr} D(\mathcal{N}(\mathbf{x}; \mu_y^{tr}, \Sigma_y^{tr})||\mathcal{N}(\mathbf{x}; \mu_y, \Sigma_y)) + \sum_y \omega_y^{tr} \ln \frac{\omega_y^{tr}}{\omega_y} - \ln \pi(f) - \beta \end{aligned} \tag{4.10}$$

Further application of Equation (2.31) proves the corollary. ▮

When $\pi(f)$ is uniform, we can discard it and renormalize $p_{\text{fid}}(f)$. Therein, the prior distribution of the mean and covariance matrix of a conditional Gaussians becomes a normal-Wishart distribution, as shown in Equation (2.31). This prior distribution has long been used in MAP adaptation of Gaussian models for speech recognition [104] due to its nice mathematical properties as a conjugate prior; here we have derived it from the fidelity prior. If $\omega_y = \omega_y^{tr}$, $\Sigma_y = \Sigma_y^{tr}$, the problem becomes to adapt Gaussian means only, and the fidelity prior becomes a joint Gaussian with a block diagonal covariance matrix on the concatenated means $[\mu_+^{tr}, \mu_-^{tr}]^T$. This implies that the accuracy-regularization optimization objective for Gaussian mean adaptation asks to minimize the negative log likelihood as well as a Mahalanobis distance from the unadapted means (a generalized ℓ_2-norm). This objective has a simple, closed-form solution since both terms are quadratic.

The above corollary can be easily extended to any class-conditional distributions that belong to the exponential family, *i.e.*, $p(\mathbf{x}|y,f) = p(\mathbf{x}|\theta_y) = \exp\{a(\mathbf{x}) + b(\theta_y) + d(\theta_y)c(\mathbf{x})\}$ where θ_y is the parameters describing the y^{tr} class-conditional distribution, and $p(y|f) = \omega_y$. Thus f is represented by (ω_y, θ_y). The KL-divergence can be conveniently calculated using Equation (2.34)

$$D(p(\mathbf{x},y|f^{tr})||p(\mathbf{x},y|f)) = \sum_y \omega_y^{tr} \left[\frac{\nabla_{\theta_y} b(\theta_y^{tr})}{\nabla_{\theta_y} d(\theta_y^{tr})}(d(\theta_y) - d(\theta_y^{tr})) - (b(\theta_y) - b(\theta_y^{tr})) \right] + \sum_y \omega_y^{tr} \ln \frac{\omega_y^{tr}}{\omega_y}$$

This is a general form applicable to many well-known distributions including the Gaussian models we discussed above.

4.2.2 Mixture models

In practice, mixture models are more useful for their ability to approximate arbitrary distributions. Mathematically, $p(\mathbf{x}|y,f) = \sum_k c_{y,k} p(\mathbf{x}|\theta_{y,k})$, where $c_{y,k}$, $k = 1..K$, are component responsibilities for class y, and $\theta_{y,k}$ are model parameters for the k^{th} component in class y. Furthermore, as in the previous case, we let $p(y|f) = \omega_y$. Thus f is represented by $(\omega_y, c_{y,k}, \theta_{y,k})$. There is no close-form solution to the KL-divergence of mixture models. However, we have the following corollary which offers a lower bound on the fidelity prior (*i.e.*, an upper bound on $-\ln p_{\text{fid}}(f)$). The same result can be found in [93, 136, 137].

Corollary 4.2.3 *For class-conditional mixture models, the prior $p_{\mathit{fid}}(f)$ defined in Equation* (4.4) *satisfies*

$$\begin{aligned} -\ln p_{\mathit{fid}}(f) \le & \sum_y \omega_y^{tr} \sum_k c_{y,k}^{tr} D(p(\mathbf{x}|\theta_{y,k}^{tr})||p(\mathbf{x}|\theta_{y,k})) \\ & + \sum_y \omega_y^{tr} \sum_k c_{y,k}^{tr} \ln \frac{c_{y,k}^{tr}}{c_{y,m(k)}} + \sum_y \omega_y^{tr} \ln \frac{\omega_y^{tr}}{\omega_y} - \ln \pi(f) - \beta \end{aligned} \tag{4.11}$$

where $\beta > 0$ and $(m(1), \ldots, m(K))$ is any permutation of $(1, \ldots, K)$.

Proof Using the log sum inequality (Theorem 2.5.2), we have

$$\begin{aligned}
& D(p(\mathbf{x}, y|f^{tr})||p(\mathbf{x}, y|f)) \\
= \; & D(p(\mathbf{x}|y, f^{tr})||p(\mathbf{x}|y, f)) + D(p(y|f^{tr})||p(y|f)) \\
= \; & \sum_y \omega_y^{tr} \int_x \sum_k \left(c_{y,k}^{tr} p(\mathbf{x}|\theta_{y,k}^{tr}) \right) \ln \frac{\sum_k c_{y,k}^{tr} p(\mathbf{x}|\theta_{y,k}^{tr})}{\sum_k c_{y,k} p(\mathbf{x}|\theta_{y,k})} \, d\mathbf{x} + \sum_y \omega_y^{tr} \ln \frac{\omega_y^{tr}}{\omega_y} \\
\leq \; & \sum_y \omega_y^{tr} \int_x \sum_k c_{y,k}^{tr} p(\mathbf{x}|\theta_{y,k}^{tr}) \ln \frac{c_{y,k}^{tr} p(\mathbf{x}|\theta_{y,k}^{tr})}{c_{y,m(k)} p(\mathbf{x}|\theta_{y,m(k)})} \, d\mathbf{x} + \sum_y \omega_y^{tr} \ln \frac{\omega_y^{tr}}{\omega_y} \\
= \; & \sum_y \omega_y^{tr} \sum_k c_{y,k}^{tr} D(p(\mathbf{x}|\theta_{y,k}^{tr})||p(\mathbf{x}|\theta_{y,m(k)})) + \sum_y \omega_y^{tr} \sum_k c_{y,k}^{tr} \ln \frac{c_{y,k}^{tr}}{c_{y,m(k)}} + \sum_y \omega_y^{tr} \ln \frac{\omega_y^{tr}}{\omega_y}
\end{aligned} \tag{4.12}$$

Applying Theorem 4.2.1 completes the proof. ∎

This corollary holds for an arbitrary alignment $m(k)$ of the mixture components. We can always choose the alignment, based on the similarity between the mixture components, that yields the minimum KL-divergence. In other words, at each EM iteration, we need to find

$$\hat{m}(\cdot) = \underset{m(\cdot)}{\text{argmin}} \left(\sum_y \omega_y^{tr} \sum_k c_{y,k}^{tr} D(p(\mathbf{x}|\theta_{y,k}^{tr})||p(\mathbf{x}|\theta_{y,m(k)})) + \sum_y \omega_y^{tr} \sum_k c_{y,k}^{tr} \ln \frac{c_{y,k}^{tr}}{c_{y,m(k)}} \right)$$

In fact, if we initialize $f = f^{tr}$ which is a common practice in adaptation, we can empirically assume that $m(k) = k$.

Moreover, The corollary implicitly assumes that K, the number of components per mixture, is fixed, but this can be easily extended to the case where K grows or shrinks during adaptation. Suppose that the adapted model is desired to have L components per mixture (where $L \neq K$), whose parameters are initialized either from the unadapted model or from the adaptation data. With other conditions unchanged, evaluating the fidelity prior is again reduced to computing $D(p(\mathbf{x}, y|f^{tr})||p(\mathbf{x}, y|f))$ as in Equation (4.12) except that the number of mixture components of $p(\mathbf{x}|y, f)$ changes from K to L. To this end, we first compute the least common multiple of K and L, denoted as M. Then we "clone" each component in f^{tr} into M/K copies, each with a scaled component responsibility $(K/M)c_{y,k}^{tr}$. Likewise, we clone each component in f into M/L copies and scale the responsibilities accordingly. In this way, the number of mixture components becomes equal, and the inequality in (4.12) still applies. Note that in adaptation, we need to *tie* the clone components of f to keep their parameters identical, and we merge them back to L components when adaptation finishes.

The above result offers an upper bound on the KL and hence a lower bound on the fidelity prior. We can replace $-\ln p_{\text{fid}}(f)$ with this lower bound in the accuracy-regularization objective in Equation (4.1) in learning mixture models. In particular, we can derive such a regularizer for Gaussian mixture models where only Gaussian means are adapted. For class-conditional Gaussian mixture models with fixed (across training and adaptation) class prior probabilities ω_y^{tr}, fixed component responsibilities $c_{y,k}^{tr}$, fixed covariance matrices $\Sigma_{y,k}^{tr}$ and a fixed alignment $m(k) = k$, the regularizer is of the form

$$\sum_y \frac{1}{2}\omega_y^{tr} \sum_k c_{y,k}(\mu_{y,k} - \mu_{y,k}^{tr})^T {\Sigma_{y,k}^{tr}}^{-1}(\mu_{y,k} - \mu_{y,k}^{tr}) - \ln \pi(f) - \beta \tag{4.13}$$

4.2.3 Hidden Markov models

A hidden Markov models (HMM) is a generative model describing two statistically dependent random processes. The first is an observable process $\{\mathbf{x}_1, \mathbf{x}_2, \dots\}$, and the second is a homogeneous hidden Markov process $\{y_1, y_2, \dots\}$, where y_t takes values in a finite state space $\mathcal{Y} = \{1, 2, \dots, n\}$. Note that we use such notation for consistency with the rest of the work. In an HMM, we have the conditional independence statements $\mathbf{x}_t \perp\!\!\!\perp \{\mathbf{x}_{1:t-1}, y_{1:t-1}\}|y_t$ and $y_{t+1} \perp\!\!\!\perp \{\mathbf{x}_{1:t-1}, y_{1:t-1}\}|y_t$. Here $A \perp\!\!\!\perp B|C$ means that A is independent of B given C. The joint likelihood of all random variables, as mentioned in Chapter 2, is given by

$$p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T, y_1, y_2, \cdots, y_T|f) = p(y_1|f)p(\mathbf{x}_1|y_1, f)\prod_{t=2}^T p(y_t|y_{t-1}, f)p(\mathbf{x}_t|y_t, f) \tag{4.14}$$

The standard parametric representation of an HMM is $f = (\pi, A, B)$. The parameter π is a *row* vector of state prior probabilities, $\pi = [\omega_1, \dots, \omega_n]$, where $\omega_i = p(y_1 = i|f)$; $A = (a_{i,j})$ is a state transition probability matrix where $a_{i,j} = p(y_{t+1} = j|y_t = i, f)$; and $B = \{b_1, \dots, b_n\}$ is a set where b_j represents the observation distribution of state j, *i.e.*, $b_j(t) \triangleq p(\mathbf{x}_t|y_t = i, f)$. We further define a number of KL-divergences defined using π, A and B respectively.

$$\begin{aligned}
D(\pi^{tr}||\pi) &\triangleq \sum_{i=1}^n \omega_i^{tr} \ln \frac{\omega_i^{tr}}{\omega_i} \\
D(a_i^{tr}||a_i) &\triangleq \sum_{j=1}^n a_{i,j}^{tr} \ln \frac{a_{i,j}^{tr}}{a_{i,j}} \\
D(b_i^{tr}||b_i) &\triangleq D(p(\mathbf{x}_t|b_i^{tr})||p(\mathbf{x}_t|b_i))
\end{aligned}$$

The *KL-divergence rate* between two HMMs is defined as

$$\lim_{T\to\infty} \frac{1}{T} D(p(\mathbf{x}_{1:T}|f^{tr})||p(\mathbf{x}_{1:T}|f)) \tag{4.15}$$

To study this quantity, we first inspect the KL-divergence w.r.t. a finite sequence of observations.

$$\begin{aligned} & D(p(\mathbf{x}_{1:T}|f^{tr})||p(\mathbf{x}_{1:T}|f)) \\ = \;& D(p(\mathbf{x}_{1:T}, y_{1:T}|f^{tr})||p(\mathbf{x}_{1:T}, y_{1:T}|f)) - D(p(y_{1:T}|\mathbf{x}_{1:T}, f^{tr})||p(y_{1:T}|\mathbf{x}_{1:T}, f)) \\ \leq \;& D(p(\mathbf{x}_{1:T}, y_{1:T}|f^{tr})||p(\mathbf{x}_{1:T}, y_{1:T}|f)) \\ = \;& D(p(\mathbf{x}_{1:T}|y_{1:T}, f^{tr})||p(\mathbf{x}_{1:T}|y_{1:T}, f)) + D(p(y_{1:T}|f^{tr})||p(y_{1:T}|f)) \end{aligned} \tag{4.16}$$

It has been proved by induction [136, 138] that the above is further upper bounded by

$$D(\pi^{tr}||\pi) + \pi^{tr}\left(\sum_{t=0}^{T-2}(A^{tr})^t\right)(d_A + d_B) + (A^{tr})^{T-1}d_B \tag{4.17}$$

where $d_A = [D(a_1^{tr}||a_1)..D(a_n^{tr}||a_n)]^T$, and $d_B = [D(b_1^{tr}||b_1)..D(b_n^{tr}||b_n)]^T$.

With modest assumptions, Equation (4.17) can be plugged into Equation (4.15) to obtain an upper bound on the KL-divergence rate of HMMs; the key is that the bound has to be well defined when $T \to \infty$. If the Markov process is stationary, *i.e.*, there exists a stationary distribution vector ν such that $\nu^T A = \nu^T$ and $\lim\limits_{t\to\infty} \pi^T A^t = \nu^T$, then [136] proved that

$$\begin{aligned} & \lim_{T\to\infty} \frac{1}{T}\Big[D(p(\mathbf{x}_{1:T}|f^{tr})||p(\mathbf{x}_{1:T}|f))\Big] \\ \leq \;& \lim_{T\to\infty} \frac{1}{T}\left[D(\pi^{tr}||\pi) + \pi^{tr}\left(\sum_{t=0}^{T-2}(A^{tr})^t\right)(d_A + d_B) + (A^{tr})^{T-1}d_B\right] \\ = \;& \pi^{tr}(d_A + d_B) \end{aligned} \tag{4.18}$$

Furthermore, [138] extended this derivation to *left-to-right* HMMs with final non-emitting states, which is a common setting in speech recognition. Under such conditions, [138] proved that the KL-divergence (rather than the KL-divergence rate) between two HMMs is well defined and is upper bounded by

$$\lim_{T\to\infty} D(p(\mathbf{x}_{1:T}|f^{tr})||p(\mathbf{x}_{1:T}|f)) \leq D(\pi^{tr}||\pi) + \pi^{tr^T}(I - A^{tr})^{-1}(d_A + d_B) \tag{4.19}$$

We can utilize this bound in our regularized adaptation objective for HMMs.

4.3 Discriminative Classifiers

Generative approaches are often suboptimal from a classification objective perspective, as they ask to solve a more difficult density estimation problem. Discriminative approaches, which directly model the conditional relationship of class label given input features, often give better classification performance. The classification decision is made via Equation (2.5) or Equation (2.6). As mentioned in Chapter 2, one class of discriminative classifiers, including MLPs, MaxEnt models and CRFs, use probabilistic models $p(y|\mathbf{x}, f)$. Although other classifiers such as kernel methods in general do not explicitly model posterior probabilities, their outputs can be given probabilistic interpretations. For example, there have been approaches to fit SVM outputs to a probability function (*e.g.* sigmoid) to enable post-processing [139]. Here we assume that $p(y|\mathbf{x}, f)$ exists in all cases.

Analogous to our discussion on generative classifiers, if we use the conditional likelihood loss $Q(\cdot) = -\ln p(y|\mathbf{x}, f)$, the unadapted model is then the *true* model that describes the conditional distribution in training, *i.e.*, $p(y|\mathbf{x}, f^{tr}) = p^{tr}(y|\mathbf{x}) = p^{tr}(\mathbf{x}, y)/p^{tr}(\mathbf{x})$; and similarly $p(y|\mathbf{x}, f^{ad}) = p^{ad}(y|\mathbf{x})$. Furthermore, the posterior probability can be expressed as

$$p(f|\mathbf{x}, y) = \frac{p(y|\mathbf{x}, f)p(f, \mathbf{x})}{p(\mathbf{x}, y)} = \frac{p(y|\mathbf{x}, f)\pi(f)}{p(y|\mathbf{x})} = \frac{p(y|\mathbf{x}, f)\pi(f)}{\int p(y|\mathbf{x}, f)\pi(f)\, df} \tag{4.20}$$

where we have assumed that f and $\mathbf{x}$ are independent random variables. Rasmussen [140] has derived the same posterior expression (see Equation (2.5) therein) under such an assumption, which is standard in most, if not all, pattern classification tasks. Note that given y, variables f and $\mathbf{x}$ are no longer independent.

This factorization leads to a result analogous to Theorem 4.2.1: Assuming that $p^{tr}(\mathbf{x}, y)$ is known, the fidelity prior for discriminative classifiers satisfies

$$-\ln p_{\text{fid}}(f) = D(p(y|\mathbf{x}, f^{tr})||p(y|\mathbf{x}, f)) - \ln \pi(f) - \beta \tag{4.21}$$

where $\beta > 0$. The proof of this can be obtained in a similar fashion:

$$\begin{aligned} -\ln p_{\text{fid}}(f) &= -\int p^{tr}(\mathbf{x}, y) \ln p(f|\mathbf{x}, y)\, d\mathbf{x}\, dy - \gamma \\ &= -\int p^{tr}(\mathbf{x}, y) \ln \left[\frac{p(f|\mathbf{x}, y)\pi(f)}{p(y|\mathbf{x}, f^{tr})} \cdot \frac{p(y|\mathbf{x}, f^{tr})}{\int p(y|\mathbf{x}, f)\pi(f)\, df} \right] d\mathbf{x}\, dy - \gamma \\ &= D(p(y|\mathbf{x}, f^{tr})||p(y|\mathbf{x}, f)) - \ln \pi(f) - D(p^{tr}(y|\mathbf{x})||p(y|\mathbf{x})) - \gamma \end{aligned} \tag{4.22}$$

Letting $\beta = D(p^{tr}(\mathbf{x}, y)||p(y|\mathbf{x})) + \gamma$, we have

$$1 = \int p_{\text{fid}}(f)\, df = \int_{\mathcal{F}} \exp\{-D(p(y|\mathbf{x}, f^{tr})||p(y|\mathbf{x}, f)) + \ln \pi(f) + \beta\}\, df < e^{\beta} \tag{4.23}$$

Therefore we have $\beta > 0$.

The training distribution $p^{tr}(\mathbf{x}, y)$, however, is generally unknown to discriminative models (the only information preserved from the training data is f^{tr} which reflects only the conditional distribution), thereby making $D(p(y|\mathbf{x}, f^{tr})||p(y|\mathbf{x}, f))$ uncomputable. The major goal of this section is to derive such an upper bound on $D(p(y|\mathbf{x}, f^{tr})||p(y|\mathbf{x}, f))$, and hence on $-\ln p_{\text{fid}}(f)$, that does not require the knowledge of $p^{tr}(\mathbf{x}, y)$. Then we can replace $-\ln p_{\text{fid}}(f)$ with this upper bound in the accuracy-regularization objective.

Many discriminative classifiers, including log linear models, conditional maximum entropy models, CRFs, MLPs and SVMs, can be viewed as hyperplane classifiers in a transformed feature space: $f(\mathbf{x}) = \text{sgn}\,\left(\mathbf{w}^T\phi(\mathbf{x}) + b\right)$, where $f = (\mathbf{w}, b)$ and $\phi(\mathbf{x})$ is a nonlinear transformation applied to the input space. In MLPs, for example, $\phi(\mathbf{x})$ is represented by hidden neurons, and in SVMs $\phi(\mathbf{x})$ is implicitly determined by a reproducing kernel. For consistency, we use $\mathbf{x}$ in this section to represent features, but $\mathbf{x}$ can be readily replaced by $\phi(\mathbf{x})$ for nonlinear cases. Moreover, for binary classification problems, a sigmoid function

$$p(y|\mathbf{x}, f) = \frac{1}{1 + e^{-y(\mathbf{w}^T\mathbf{x}+b)}} \tag{4.24}$$

is often used to model conditional distributions for such classifiers (while a softmax function is often used for the multi-class case). Plugging Equation (4.24) into Equation (4.21), we arrive at the following theorem.

Theorem 4.3.1 *For hyperplane classifiers sgn* $(\mathbf{w}^T\mathbf{x} + b)$*, the fidelity prior in Equation* (4.21) *satisfies*

$$-\ln p_{\textit{fid}}(f) \le \alpha\|\mathbf{w} - \mathbf{w}^{tr}\| + |b - b^{tr}| - \ln \pi(f) - \beta \tag{4.25}$$

where $\alpha = \mathrm{E}_{\mathbf{x}\sim p^{tr}(\mathbf{x})}[\|\mathbf{x}\|]$.

Proof First we show that for any $a > 0$ and $b > 0$, $|\ln \dfrac{1+a}{1+b}| \le |\ln a - \ln b|$. This is because

$$\begin{cases} 1 \le \dfrac{1+a}{1+b} \le \dfrac{a}{b} & \text{if } a \ge b \\ \dfrac{a}{b} < \dfrac{1+a}{1+b} < 1 & \text{otherwise} \end{cases} \tag{4.26}$$

Taking logarithm on both sides will prove the above inequality. Utilizing this inequality and the fact that $e^x > 0$ for any x, we have

$$\begin{aligned} D(p(y|\mathbf{x}, f^{tr})||p(y|\mathbf{x}, f)) &= \int p^{tr}(\mathbf{x}, y) \ln \frac{1 + e^{-y(\mathbf{w}^T\mathbf{x}+b)}}{1 + e^{-y(\mathbf{w}^{tr^T}\mathbf{x}+b^{tr})}}\, d\mathbf{x}\, dy \\ &\leq \int p^{tr}(\mathbf{x}, y) |\ln e^{-y(\mathbf{w}^T\mathbf{x}+b)} - \ln e^{-y(\mathbf{w}^{tr}+b^{tr})}|\, d\mathbf{x}\, dy \\ &= \int p^{tr}(\mathbf{x}, y) |y(\mathbf{w}^{tr} - \mathbf{w})^T\mathbf{x} + y(b^{tr} - b)|\, d\mathbf{x}\, dy \\ &\leq \int p^{tr}(\mathbf{x}, y) |y(\mathbf{w}^{tr} - \mathbf{w})^T\mathbf{x}|\, d\mathbf{x}\, dy + \int p^{tr}(\mathbf{x}, y) |y(b^{tr} - b)|\, d\mathbf{x}\, dy \\ &= ||\mathbf{w} - \mathbf{w}^{tr}|| \int p^{tr}(\mathbf{x}) ||\mathbf{x}||\, d\mathbf{x} + |b - b^{tr}| \\ &= \alpha ||\mathbf{w} - \mathbf{w}^{tr}|| + |b - b^{tr}| \quad \blacksquare \end{aligned} \tag{4.27}$$

Therefore, the accuracy-regularization optimization objective becomes

$$\min_{\mathbf{w},b} R_{\text{emp}}(\mathbf{w}, b) + \frac{\lambda_1'}{2} ||\mathbf{w} - \mathbf{w}^{tr}|| + \frac{\lambda_2'}{2} |b - b^{tr}| - \lambda_2' \ln \pi(\mathbf{w}, b) \tag{4.28}$$

where λ_1' and λ_2' are regularization coefficients. Assuming that $\pi(\mathbf{w}, b)$ is uniform (only for simplicity), this optimization objective asks to minimize the empirical risk as well as the ℓ_2-norms. Notice that for a random variable $\mathbf{x}$, $||\mathbf{x}||$ is convex (not necessarily strictly) in $\mathbf{x}$ and is differentiable everywhere except $\mathbf{x} = \mathbf{0}$. In practice, it is often more convenient to use the square of the ℓ_2-norm in the optimization objective as follows,

$$\min_{\mathbf{w},b} R_{\text{emp}}(\mathbf{w}, b) + \frac{\lambda_1}{2} ||\mathbf{w} - \mathbf{w}^{tr}||^2 + \frac{\lambda_2}{2} |b - b^{tr}|^2 - \lambda_2 \ln \pi(\mathbf{w}, b). \tag{4.29}$$

Since $||\mathbf{x}||^2$ is strictly convex and is differentiable everywhere, using the ℓ_2 norm square gives great mathematical convenience. Moreover, considering the problem of minimizing Equation (4.28) and (4.29) with the assumption that $R_{\text{emp}}(\mathbf{w}, b)$ is twice differentiable in $(\mathbf{w}, b)$, we can in fact prove that for any choice of λ_1' and λ_2' in Equation (4.28), there always exist λ_1 and λ_2 in Equation (4.29) such that a locally optimal solution to (4.28) is also a locally optimal solution to (4.29). To see this, it is sufficient to prove the following lemma.

Lemma 4.3.2 *For a twice differentiable function $g(\mathbf{x})$. Consider the problem of minimizing J_1 and J_2 defined as*

$$\begin{aligned} J_1(\mathbf{x}) &= g(\mathbf{x}) + \frac{\lambda'}{2} ||\mathbf{x}|| \\ J_2(\mathbf{x}) &= g(\mathbf{x}) + \frac{\lambda}{2} ||\mathbf{x}||^2 \end{aligned} \tag{4.30}$$

For any $\lambda' > 0$ in J_1, there always exists $\lambda > 0$ in J_2 such that a locally optimal solution to $J_1(\mathbf{x})$ is also a locally optimal solution to $J_2(\mathbf{x})$.

The proof is provided in Appendix A.6. Replacing $\mathbf{x}$ with $(\mathbf{w}, b)$, and replacing $g(\mathbf{x})$ with $R_{\text{emp}}(\mathbf{w}, b)$, we reach the earlier conclusion.

Before we evaluate regularized adaptation algorithms in Chapter 5, we derive generalization error bounds for adaptation in the PAC-Bayesian framework.

4.4 PAC-Bayesian Error Bound Analysis

As described in Chapter 2, a fundamental problem in machine learning is to study the generalization performance of a classifier in terms of an error bound or, equivalently, a sample complexity bound. A PAC-Bayesian approach [60] incorporates domain knowledge in the form of a Bayesian prior and provides a guarantee on generalization error regardless of the truth of the prior. In this chapter, we are particularly interested in how well an adapted classifier generalizes to unseen data drawn from the target distribution. We derive error bounds for adaptation by using our proposed prior in the PAC-Bayesian setting. Specifically, for a countable function space, we apply the Occam's Razor bound (Theorem 2.3.3) which bounds the true error of a single classifier; while for a continuous function space, we apply McAllester's PAC-Bayes bound (Theorem 2.3.5) which bounds the true stochastic error of a Gibbs classifier.

There is one issue we need to clarify before moving on. All PAC-Bayesian theorems [60] assume a finite loss function, *i.e.* $Q(\cdot) \in [a, b]$, in order to utilize Hoeffding inequality or its alternatives [60, 85, 87]. Moreover, $Q(\cdot) \in [0, 1]$ is often used instead since any $Q(\cdot) \in [a, b]$ can be shifted and scaled to be in this interval . In Section 2.3, we particularly assumed that $Q(f(\mathbf{x}), y) = \mathrm{I}(f(\mathbf{x}) \neq y)$, and we keep this assumption throughout this section. In other words, we have

$$\begin{aligned} R_{p^{ad}(\mathbf{x},y)}(f) &= \mathrm{E}_{p^{ad}}(\mathbf{x}, y)[I(f(\mathbf{x}) \neq y)] \\ R_{\text{emp}}(f) &= \frac{1}{m}\sum_{i=1}^{m} I(f(\mathbf{x}_i) \neq y_i);\ (\mathbf{x}_i, y_i) \in \mathcal{D}_m^{ad} \end{aligned} \tag{4.31}$$

PAC-Bayesian theorems (see Chapter 2) can be utilized to upper bound $R_{p^{ad}(\mathbf{x},y)}(f)$ by $R_{\text{emp}}(f)$ plus a capacity term $\Phi(\mathcal{F}, f, \mathcal{D}_m, m, \delta)$, *i.e.*,

$$R_{p^{ad}(\mathbf{x},y)}(f) \leq R_{\text{emp}}(f) + \Phi(\mathcal{F}, f, \mathcal{D}_m, m, \delta)$$

Note that f can be estimated using a different loss function, but the above bound on the expected 0-1 loss is *valid regardless of the loss function used in training*. In practice, surrogates of the 0-1 loss are often used, as there exists a quantitative relationship between the risk as accessed using the 0-1 loss and the risk as accessed using the surrogates [57].

4.4.1 Occam's Razor bound for adaptation

The Occam's Razor bound (Theorem 2.3.3) implies that, in order to guarantee a small true error, we should intuitively assign high prior probabilities to those models which are a-priori viewed as likely to fit the data well. The use of the fidelity prior in adaptation follows this intuition. To derive a generalization error bound for adaptation, we replace the standard prior $\pi(f)$ in Equation (2.19) by our proposed fidelity prior $p_{\text{fid}}(f)$. We in particular study a countable function space of generative models in this subsection.

Corollary 4.4.1 *For a countable function space of generative models, for any prior distribution $\pi(f)$ and for any f for which $\pi(f) > 0$, the following bound holds with probability of at least $1 - \delta$,*

$$R_{p^{ad}}(f) \leq R_{emp}(f) + \sqrt{\frac{D(p(\mathbf{x}, y|f^{tr})||p(\mathbf{x}, y|f)) - \ln \pi(f) - \beta - \ln \delta}{2m}} \tag{4.32}$$

This result has important implications: for the set of classifiers

$$\mathcal{G} = \{f \in \mathcal{F} : D(p(\mathbf{x}, y|f^{tr})||p(\mathbf{x}, y|f)) < \beta\},$$

their error bounds in Equation (4.32) which use the fidelity prior are tighter than those in Equation (2.19) which use the standard prior. Since $\beta > 0$, $\mathcal{G}$ is always nonempty. For classifiers in the complementary set $\bar{\mathcal{G}}$, however, we reach the opposite argument. An important question to ask is: to which set does our estimated classifier belongs? We are particularly interested in f^{ad}, *i.e.*, the optimal classifier w.r.t. the target distribution. If $D(p^{tr}||p^{ad}) < \beta$, we have $f^{ad} \in \mathcal{G}$ and we achieve better generalization performance at f^{ad} by using the fidelity prior. Practically speaking, this implies that it is better to utilize adaptation unless the training and target distributions are different to a certain extent (determined by β). Recall that β normalizes $p_{\text{fid}}(f)$ to unity. This constant can be analytically calculated for some models (e.g. Gaussian models), while approximations are needed for general cases. Additionally, we can derive a similar bound for discriminative classifiers, where the divergence in Equation (4.32) is between conditional distributions instead of joint distributions.

4.4.2 PAC-Bayesian bounds for adaptation

McAllester's PAC-Bayesian bound for Gibbs classifiers (Theorem 2.3.5) is applicable to both countable and uncountable function spaces. A Gibbs classifier is a stochastic classifier drawn from a posterior distribution $q(f)$. Consequently the true error and empirical error also become stochastic in the form of $\mathrm{E}_{f\sim q(f)}[R(f)]$ and $\mathrm{E}_{f\sim q(f)}[R_{\text{emp}}(f)]$. Again, the choice of a prior distribution $\pi(f)$ is critical in order to achieve a small error bound. Intuitively we should choose a distribution $\pi(f)$ such that $\mathrm{E}_{f\sim\pi(f)}[R_{\text{emp}}(f)]$ is small. As a ramification of this theorem, *PAC-Bayesian margin bounds* have been developed which provide theoretical foundations for SVMs [92]. The key idea involves choosing a prior $\pi(f)$ and a posterior $q(f)$ such that, in addition to our intuition above, it is easy to compute $D(q(f)||\pi(f))$ and $\mathrm{E}_{f\sim q(f)}[R_{\text{emp}}(f)]$. Usually $q(f)$ is chosen to be in the same family as $\pi(f)$.

In this section, we obtain error bounds for adaptation in a similar fashion as [92] but with simpler derivations. Since the derivation requires specification of a classifier, we first investigate generative Gaussian models where only Gaussian means are adapted. We further assume equal class prior probabilities ω_+=ω_-=$1/2$, equal covariance matrices Σ_+=Σ_-=Σ^{tr}, and opposite means μ_+=$-\mu_-$=μ, thereby leading to a linear decision boundary $f(\mathbf{x}) = \text{sgn}\ ({\Sigma^{tr}}^{-1}\mu^T\mathbf{x})$. In such a case, f is represented by μ only. We make such assumptions only to simplify the calculation of the stochastic error in this work, while similar bounds can be derived for more general cases. Next, we present a corollary of Theorem 2.3.5 followed by a proof and discussions.

Corollary 4.4.2 *(***PAC-Bayesian bound for symmetric Gaussian model adaptation***) Assume class-conditional Gaussian models with equal and fixed class prior probabilities $\omega^{tr} = 1/2$, equal and fixed covariance matrices Σ^{tr}, and opposite means $\mu_+ = -\mu_- = \mu$. Define an approximate posterior distribution on μ with the form $q(\mu) = \mathcal{N}(\mu;\ \mu', \Sigma^{tr})$. Then there exists a prior distribution such that for any μ', the following bound holds true with probability of at least $1-\delta$.*

$$\mathrm{E}_{q(\mu)}[R_{p^{ad}}(\mu)] \leq \frac{1}{m}\sum_{i=1}^{m} F\left(\frac{y_i\mathbf{x}_i^T{\Sigma^{tr}}^{-1}\mu'}{(\mathbf{x}_i^T{\Sigma^{tr}}^{-1}\mathbf{x}_i)^{1/2}}\right)+\sqrt{\frac{\frac{1}{2}(\mu'-\mu^{tr})^T{\Sigma^{tr}}^{-1}(\mu'-\mu^{tr}) - \ln\delta + \ln m + 2}{2m-1}} \tag{4.33}$$

where $F(t) = \int_t^\infty \frac{1}{\sqrt{2\pi}}e^{-\frac{s^2}{2}}\,ds$ and $(\mathbf{x}_i, y_i) \in \mathcal{D}_m^{ad}$.

Proof McAllester's PAC-Bayesian bound allows us to choose any prior distribution. Here we use $p_{\text{fid}}(f)$ in Equation (4.4), where we assume a uniform $\pi(f)$ and renormalize $p_{\text{fid}}(f)$ accordingly. This results in a Gaussian distribution on μ, *i.e.*, $p_{\text{fid}}(\mu) = \mathcal{N}(\mu; \mu^{tr}, \Sigma^{tr})$. Furthermore, by definition $q(\mu) = \mathcal{N}(\mu;\ \mu', \Sigma^{tr})$. It is easy to compute the KL-divergence

$$\begin{aligned} D(q(\mu)\|p_{\text{fid}}(\mu)) &= D(\mathcal{N}(\mu;\ \mu', \Sigma^{tr})\|\mathcal{N}(\mu;\ \mu^{tr}, \Sigma^{tr})) \\ &= \frac{1}{2}(\mu' - \mu^{tr})^T {\Sigma^{tr}}^{-1}(\mu' - \mu^{tr}) \end{aligned} \tag{4.34}$$

which gives the second term in Equation (4.33).

On the other hand, to calculate $\mathrm{E}_{f\sim q(f)}[R_{\text{emp}}(f)]$), we first inspect the decision function regarding sample $(\mathbf{x}_i, y_i)$, *i.e.*,

$$\text{sgn}\left(y_i(\mu_+ - \mu_-)^T {\Sigma^{tr}}^{-1}\mathbf{x}_i\right) = \text{sgn}\left(y_i \mathbf{x}_i^T {\Sigma^{tr}}^{-1}\mu\right).$$

Since an affine transformation of a multivariate Gaussian is still a Gaussian, $y_i \mathbf{x}_i^T {\Sigma^{tr}}^{-1}\mu$ is a univariate Gaussian with the following mean and variance:

$$\begin{aligned} \bar{\mu}_i &\triangleq y_i \mathbf{x}_i^T {\Sigma^{tr}}^{-1}\mu' \\ \bar{\sigma}_i^2 &\triangleq (y_i \mathbf{x}_i^T {\Sigma^{tr}}^{-1})\Sigma^{tr}(y_i \mathbf{x}_i^T {\Sigma^{tr}}^{-1})^T = \mathbf{x}_i^T {\Sigma^{tr}}^{-1}\mathbf{x}_i \end{aligned} \tag{4.35}$$

This is the key difference from the derivation of [92]. The stochastic empirical error hence becomes

$$\begin{aligned} \mathrm{E}_{f\sim q(f)}[R_{\text{emp}}(f)] &= \frac{1}{m}\sum_{i=1}^{m} \mathrm{E}_{\mu\sim\mathcal{N}(\mu;\mu',\Sigma^{tr})}[\mathrm{I}(y_i \mathbf{x}_i^T {\Sigma^{tr}}^{-1}\mu < 0)] \\ &= \frac{1}{m}\sum_{i=1}^{m} \mathrm{E}_{t\sim\mathcal{N}(t;\bar{\mu}_i,\bar{\sigma}_i^2)}[\mathrm{I}(t < 0)] \\ &= \frac{1}{m}\sum_{i=1}^{m} F\left(\frac{\bar{\mu}_i}{\bar{\sigma}_i}\right) \end{aligned} \tag{4.36}$$

where $F(t) = \int_t^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{s^2}{2}}\, ds$, and $(\mathbf{x}_i, y_i) \in \mathcal{D}_m^{ad}$. ∎

Next we remark on the role of μ': Given the Gaussian assumption in Equation (4.37), $q(\mu)$ is solely determined by hyperparameter μ'. The learning problem, therefore, is reduced to the estimation of μ', which is a tradeoff between fitting the adaptation data, via reducing $F\left(\frac{\bar{\mu}_i}{\bar{\sigma}_i}\right)$, and staying in vicinity of the unadapted parameter μ^{tr}.

Lastly, we derive an adaptation error bound for hyperplane classifiers, which is an important representative for discriminative classifiers (see Section 4.3). In this case, $f = (\mathbf{w}, b)$ where we assume that $\mathbf{w}$ and b are independent variables.

Corollary 4.4.3 *(***PAC-Bayesian bound for hyperplane adaptation***) Consider hyperplane classifiers $\mathbf{w}^T\mathbf{x} + b$, and define an approximate posterior distribution on $(\mathbf{w}, b)$ with the form*

$$q(\mathbf{w}, b) = \mathcal{N}(\mathbf{w};\ \mathbf{w}', \Sigma^{tr})\mathcal{N}(b;\ b', \Sigma^{tr}) \tag{4.37}$$

Then there exists a prior distribution such that for any $(\mathbf{w}', b')$, the following bound holds true with probability of at least $1 - \delta$.

$$\mathrm{E}_{q(\mathbf{w},b)}[R_{p(\mathbf{x},y)}(f)] \leq \frac{1}{m}\sum_{i=1}^{m} F\left(\frac{y_i(\mathbf{x}_i^T\mathbf{w}' + b')}{\sqrt{\|\mathbf{x}_i\|^2 + 1}}\right) + \sqrt{\frac{\frac{\|\mathbf{w}' - \mathbf{w}^{tr}\|^2 + |b' - b^{tr}|^2}{2} - \ln\delta + \ln m + 2}{2m - 1}} \tag{4.38}$$

where $F(t) = \int_t^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{s^2}{2}}\, ds$, and $(\mathbf{x}_i, y_i) \in \mathcal{D}_m^{ad}$.

Proof We use a Gaussian prior $p(\mathbf{w}, b)$ centered at $(\mathbf{w}^{tr}, b^{tr})$ with an identity covariance matrix. Note that the choice of this prior relates to previous work on margin bounds; [92] used a Gaussian prior centered at zero, and [141] estimated Gaussian priors based on previous training subsets. The key difference is that we choose a Gaussian centered at the unadapted parameters. Mathematically, $p(\mathbf{w}, b) = \mathcal{N}(\mathbf{w}; \mathbf{w}^{tr}, I) \cdot \mathcal{N}(b; b^{tr}, 1)$, and $q(\mathbf{w}, b) = \mathcal{N}(\mathbf{w}; \mathbf{w}', I) \cdot \mathcal{N}(b; b', 1)$. The KL-divergence of two such Gaussian distributions is easily obtained as

$$D(q(f)||p(f)) = \frac{\|\mathbf{w}' - \mathbf{w}^{tr}\|^2 + |b' - \hat{b}|^2}{2} \tag{4.39}$$

Secondly, the stochastic empirical risk can be calculated similar to the Gaussian model case.

$$\begin{aligned}
\mathrm{E}_{f\sim q(f)}[R_{\text{emp}}(f)] &= \frac{1}{m}\sum_{i=1}^{m} \mathrm{E}_{\mathbf{w}\sim\mathcal{N}(\mathbf{w};\mathbf{w}',I)}\, \mathrm{E}_{b\sim\mathcal{N}(b;b',1)}[\mathrm{I}(y_i(\mathbf{w}^T\mathbf{x}_i + b) < 0)] \\
&= \frac{1}{m}\sum_{i=1}^{m} \mathrm{E}_{t\sim\mathcal{N}(y_i(\mathbf{x}_i^T\mathbf{w}'+b'), \|\mathbf{x}_i\|^2+1)}[\mathrm{I}(t < 0)] \\
&= \frac{1}{m}\sum_{i=1}^{m} F\left(\frac{y_i(\mathbf{x}_i^T\mathbf{w}' + b')}{\sqrt{\|\mathbf{x}_i\|^2 + 1}}\right)
\end{aligned} \tag{4.40}$$

where $F(t) = \int_t^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{s^2}{2}}\, ds$, and $(\mathbf{x}_i, y_i) \in \mathcal{D}_m^{ad}$. ▮

Similarly to the last corollary, $q(\mathbf{w}, b)$ is solely determined by hyperparameters $\mathbf{w}'$ and b', the estimation of which is a tradeoff between the goodness of data fitting and the fidelity to the unadapted model.

4.4.3 A VC perspective

The adaptation can also be viewed from a structured risk minimization perspective. Recall that structured risk minimization asks to search through a sequence of function spaces $\mathcal{F}_1 \subset \mathcal{F}_2 \subset ... \subset \mathcal{F}$, and find an index j that minimizes Equation (2.25). Finding an optimal $\mathcal{F}_j$ is a tradeoff between reducing the VC dimension and reducing the empirical risk; choosing a small j will decrease, or at least not increase, the VC dimension at the cost of a potential increase in the empirical risk. The question is how to design the structure of $\mathcal{F}_j$, $j = 1, 2, \ldots$, such that the increase in the empirical risk is maximally suppressed. A natural strategy is to choose a set of functions that all include the unadapted model, *i.e.*, $f^{tr} \in \mathcal{F}_j$ for all j. This is because f^{tr} minimizes the expected risk w.r.t. the training distribution, and it may as well yield a low expect risk w.r.t. the target distribution, provided that these two distributions do not diverge too much.

Consider linear classifiers $f(\mathbf{x}) = \text{sgn}\ (\mathbf{w}^T\mathbf{x})$ in canonical form with respect to a set of inputs $\{\mathbf{x}_i\}_{i=1}^m$, i.e.,

$$\min_i |\mathbf{w}^T\mathbf{x}_i| = 1$$

We can construct a sequence of constrained function spaces $\|\mathbf{w} - \mathbf{w}^{tr}\|^2 \le c_i$, where $c_1 < c_2 < ... < \infty$. All $\mathcal{F}_i$ in this sequence satisfy $\mathbf{w}^{tr} \in \mathcal{F}_i$. We have the following corollary derived from Theorem 2.4.1

Corollary 4.4.4 *Consider linear classifiers $f(\mathbf{x}) = sgn\ (\mathbf{w}^T\mathbf{x})$ such that $\min_i |(\mathbf{w} - \mathbf{w}^{tr})^T\mathbf{x}_i| = 1$ for all $\{\mathbf{x}_i\}_{i=1}^m$. For any $\mathbf{w}^{tr}$, the decision function f satisfying the constraint $\|\mathbf{w} - \mathbf{w}^{tr}\| \le c$ has a VC dimension satisfying*

$$h \le R^2(\|\mathbf{w}^{tr}\| + c)^2$$

where $R = \max_i \|\mathbf{x}_i\|$.

A proof of this corollary can be found in Appendix A.4, which makes a slight modification to Appendix A.3 (Schölkopf's proof of VC dimension). Moreover, if we use linear classifiers $(\mathbf{w} - \mathbf{w}^{tr})^T\mathbf{x} = 0$ instead of $\mathbf{w}^T\mathbf{x} = 0$ with the constraint that $\|\mathbf{w} - \mathbf{w}^{tr}\|^2 \le c$, we can achieve the same upper bound on VC dimension as in Theorem 2.4.1.

Corollary 4.4.5 *Consider linear classifiers $f(\mathbf{x}) = sgn\ ((\mathbf{w} - \mathbf{w}^{tr})^T\mathbf{x})$ such that $\min_i |(\mathbf{w} - \mathbf{w}^{tr})^T\mathbf{x}_i| = 1$ for all $\{\mathbf{x}_i\}_{i=1}^m$. For any $\mathbf{w}^{tr}$, the decision function f satisfying the constraint*

$\|\mathbf{w} - \mathbf{w}^{tr}\| \leq c$ *has a VC dimension satisfying*

$$h \leq R^2 c^2$$

where $R = \max_i \|\mathbf{x}_i\|$.

Again, a proof is provided in Appendix A.5 as a reference.

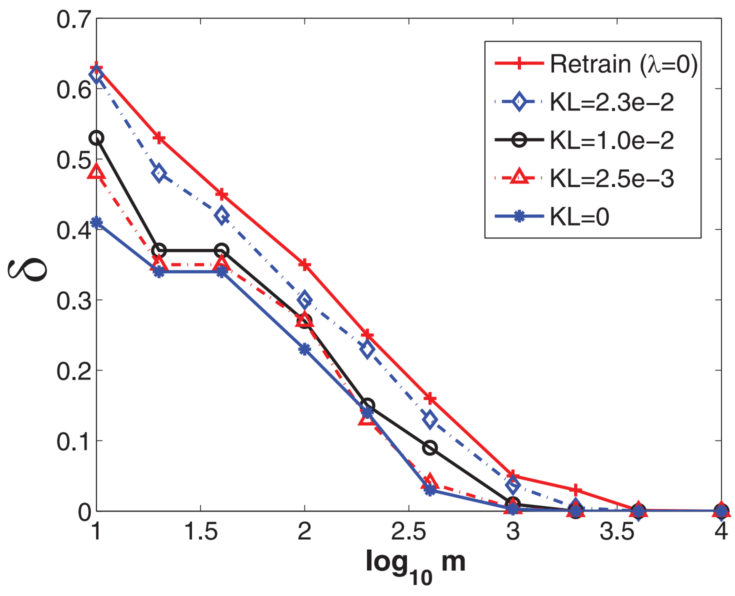
4.5 Empirical Simulations of Adaptation Error Bounds

We simulated *empirical* adaptation error bounds for a Gaussian model classifier. Given an unadapted model, and an adaptation set with m samples randomly generated from a target distribution, we learned an adapted classifier using our regularized adaptation objective in Equation (4.7), where the log joint likelihood loss and a uniform $\pi(f)$ were used, and where λ's for different m's were discovered using a development set with 5,000 samples. We computed the empirical error $R_{\text{emp}}(f)$ on the adaptation set, and estimated the true error $R(f)$ on a testing set with 10,000 samples (both corresponding to the 0-1 loss). We then estimated

$$\delta = E[\mathrm{I}(R(f) > R_{\text{emp}}(f) + \epsilon)]$$

using 1,000 separate runs (10,000 samples each).

Figure 4.1 plots δ vs. $\log m$ for $\epsilon = 0.02$ and $\epsilon = 0.1$ with different $D(p^{tr}||p^{ad})$ and m on simulated 2D-Gaussians. The $\lambda = 0$ line corresponds to retraining from scratch (no adaptation), and also to large KL-divergences, as then optimal λ discovery produces $\lambda = 0$. Although we do not yet have a theoretical result to bound $R(f)$ by $R_{\text{emp}}(f)$ in the Gaussian model case, as the function space is continuous, we have empirically shown that fewer samples were needed for smaller KL values to achieve the same confidence δ.



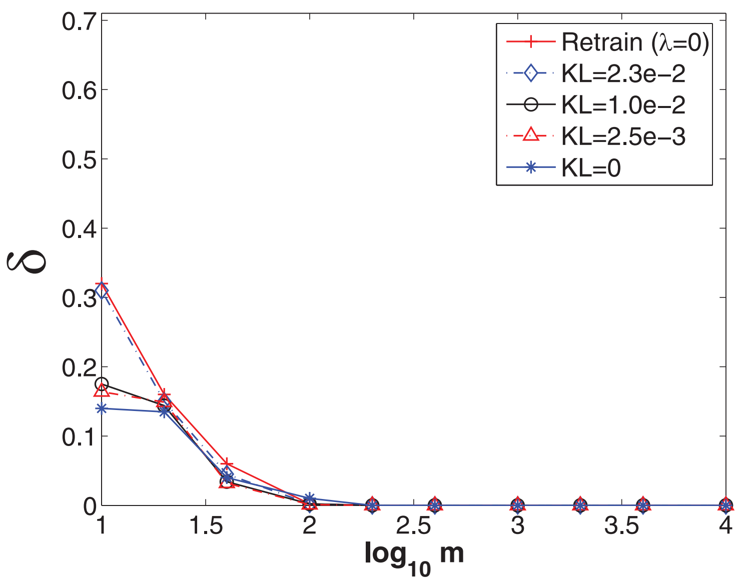


Figure 4.1: Empirical error bound study: δ vs. $\log m$ for $\epsilon = 0.02$ (upper figure) and $\epsilon = 0.1$ (lower figure)

Chapter 5
REGULARIZED ADAPTATION ALGORITHMS

In Chapter 4, we proposed a general adaptation objective, *i.e.*, $\min_{f \in \mathcal{F}} R_{\text{emp}}(f) - \lambda \ln p_{\text{fid}}(f)$, where f represents model parameters and $p_{\text{fid}}(f)$ is the fidelity prior. In practice, while optimizing this objective is often intractable, we can minimize its upper bounds instead, *e.g.* using Equation (4.11) for mixture models and Equation (4.25) for log linear models. This chapter discusses the instantiations of these algorithms for Gaussian mixture models (GMMs), support vector machines (SVMs) and multi-layer perceptrons (MLPs) (Section 5.1 through Section 5.4), with focus on the last two classifiers as they have received relatively less attention on adaptation techniques (see our discussion in Chapter 1). Furthermore, in Section 5.5, we briefly discuss the relationship between adaptation and inverse problems studied in the field of combinatorial optimization. Finally, we present MLP and SVM adaptation experiments on a vowel classification dataset (for speaker adaptation) and an object recognition dataset (for lighting condition adaptation).

5.1 GMM Adaptation

A general strategy for adapting generative models is given by Equation (4.7) in Chapter 4, where we use the joint likelihood loss $Q(f(\mathbf{x}), y) = -\ln p(\mathbf{x}, y|f)$ to compute the empirical risk on the adaptation data, and we regularize using the fidelity prior presented in Equation (4.4). In particular, we apply this "regularized adaptation" algorithm to Gaussian mixture models, where $p(\mathbf{x}|y, f) = \sum_k c_{y,k}\mathcal{N}(\mathbf{x}; \mu_{y,k}, \Sigma_{y,k})$, and $p(y|f) = \omega_y$ (following our notation in Chapter 4).

Here we investigate the case where the adaptation is partially supervised. In other words, for each input $\mathbf{x}_i$, the class membership y_i is present but the component membership k_i is missing. Consequently, the computation of the empirical risk involves marginalizing the complete likelihood over k_i before taking logarithm, *i.e.*

$$R_{\text{emp}}(f) = -\frac{1}{m}\sum_{i=1}^{m} \ln p(\mathbf{x}_i, y_i|f) = -\frac{1}{m}\sum_{i=1}^{m} \ln \sum_k p(\mathbf{x}_i, y_i, k_i = k|f) \tag{5.1}$$

which can be not optimized directly. We can, however, iteratively minimize an upper bound of this using the expectation-maximization (EM) algorithm [142]. Specifically, we let $\delta_y(i) \triangleq \mathrm{I}(y_i = y)$ denote an indicator function which equals one only when $\mathbf{x}_i$ belongs to class y, and let $L_{k|y}(i) \triangleq p(k_i = k|\mathbf{x}_i, y_i = y, f^g)$ denote the component occupancy probability of sample $(\mathbf{x}_i, y_i = y)$ which is calculated using model parameter in the previous update. Then an upper bound of the empirical risk can be derived using Jensen's inequality:

$$\begin{aligned} R_{\text{emp}}(f) &= -\frac{1}{m}\sum_{i=1}^{m}\sum_{y}\delta_y(i)\ln\sum_{k}p(\mathbf{x}_i, y_i = y, k_i = k|f) \\ &= -\frac{1}{m}\sum_{i=1}^{m}\sum_{y}\delta_y(i)\ln\sum_{k}\frac{L_{k|y}(i)p(\mathbf{x}_i, y_i = y, k_i = k|f)}{L_{k|y}(i)} \\ &\leq -\frac{1}{m}\sum_{i=1}^{m}\sum_{y}\delta_y(i)\sum_{k}L_{k|y}(i)\ln\frac{p(\mathbf{x}_i, y_i = y, k_i = k|f)}{L_{k|y}(i)} \\ &= -\sum_{y}\sum_{k}\frac{1}{m}\sum_{i=1}^{m}\delta_y(i)L_{k|y}(i)\ln p(\mathbf{x}_i, y_i = y, k_i = k|f) + C \end{aligned} \tag{5.2}$$

where C is a constant independent of the model parameters to be adapted. Furthermore, the complete likelihood can be factorized as

$$\begin{aligned} p(\mathbf{x}_i, y_i = y, k_i = k|f) &= p(y_i = y)p(k_i = k|y_i = y)p(\mathbf{x}_i|k_i = k, y_i = y) \\ &= \omega_y c_{y,k}\mathcal{N}(\mathbf{x}_i; \mu_{y,k}, \Sigma_{y,k}) \end{aligned} \tag{5.3}$$

which decomposes the adaptation objective into several independent optimization problems.

$$\begin{aligned} R_{\text{emp}}(f) \leq & -\sum_{y}\frac{1}{m}\sum_{i=1}^{m}\delta_y(i)\ln\omega_y - \sum_{y}\sum_{k}\frac{1}{m}\sum_{i=1}^{m}\delta_y(i)L_{k|y}(i)\ln c_{y,k} \\ & -\sum_{y}\sum_{k}\frac{1}{m}\sum_{i=1}^{m}\delta_y(i)L_{k|y}(i)\ln\mathcal{N}(\mathbf{x}_i; \mu_{y,k}, \Sigma_{y,k}) + C \end{aligned} \tag{5.4}$$

On the other hand, Corollary 4.2.3 provides a bound on the fidelity prior for GMMs. Exacting out the terms independent of the model parameters, and assuming a uniform standard prior $\pi(f)$, we have

$$\begin{aligned} & -\ln p_{\text{fid}}(f) \leq -\sum_{y}\omega_y^{tr}\ln\omega_y - \sum_{y}\omega_y^{tr}\sum_{k}c_{y,k}^{tr}\ln c_{y,k} \\ & +\sum_{y}\omega_y^{tr}\sum_{k}c_{y,k}^{tr}D(\mathcal{N}(\mathbf{x}; \mu_{y,k}^{tr}, \Sigma_{y,k}^{tr})||\mathcal{N}(\mathbf{x}_i; \mu_{y,k}, \Sigma_{y,k})) + C' \end{aligned} \tag{5.5}$$

Combining these two bounds, we obtain an upper bound on our regularized adaptation objective $R_{\text{emp}}(f) - \lambda\ln p_{\text{fid}}(f)$, which is convex in all model parameters. The derivation of EM update equations then becomes straightforward. At each E-step, we compute $L_{k|y}(i)$ using the current model parameters f^g,

$$L_{k|y}(i) = \frac{\omega_y^g c_{y,k}^g \mathcal{N}(\mathbf{x}_i; \mu_{y,k}^g, \Sigma_{y,k}^g)}{\sum_k \omega_y^g c_{y,k}^g \mathcal{N}(\mathbf{x}_i; \mu_{y,k}^g, \Sigma_{y,k}^g)} \tag{5.6}$$

and accumulate sufficient statistics required by the M-step updates; and at each M-step, we re-estimate the model parameters as follows,

$$\begin{aligned}
\hat{\omega}_y &= \frac{\frac{1}{m}\sum_{i=1}^m \delta_y(i) + \lambda\omega_y^{tr}}{1+\lambda} \\
\hat{c}_{y,k} &= \frac{\frac{1}{m}\sum_{i=1}^m \delta_y(i)L_{k|y}(i) + \lambda\omega_y^{tr}c_{y,k}^{tr}}{\frac{1}{m}\sum_{i=1}^m \delta_y(i) + \lambda\omega_y^{tr}} \\
\hat{\mu}_{y,k} &= \frac{\frac{1}{m}\sum_{i=1}^m \delta_y(i)L_{k|y}(i)\mathbf{x}_i + \lambda\omega_y^{tr}c_{y,k}^{tr}\mu_{y,k}^{tr}}{\frac{1}{m}\sum_{i=1}^m \delta_y(i)L_{k|y}(i) + \lambda\omega_y^{tr}c_{y,k}^{tr}} \\
\hat{\Sigma}_{y,k} &= \frac{\frac{1}{m}\sum_{i=1}^m \delta_y(i)L_{k|y}(i)(\mathbf{x}_i-\hat{\mu}_{y,k})(\mathbf{x}_i-\hat{\mu}_{y,k})^T + \lambda\omega_y^{tr}c_{y,k}^{tr}\left(\Sigma_{y,k}^{tr} + (\mu_{y,k}^{tr}-\hat{\mu}_{y,k})(\mu_{y,k}^{tr}-\hat{\mu}_{y,k})^T\right)}{\frac{1}{m}\sum_{i=1}^m \delta_y(i)L_{k|y}(i) + \lambda\omega_y^{tr}c_{y,k}^{tr}}
\end{aligned} \tag{5.7}$$

A derivation of the M-step updates is presented in Appendix B.1. These solutions are essentially the same as the MAP estimates presented in [104], as we have shown in Chapter 4 that the fidelity prior for GMMs is equivalent to the conjugate prior for GMMs with an appropriate setting of hyperparameters.

Furthermore, for the estimation of $c_{y,k}$, $\mu_{y,k}$ and $\Sigma_{y,k}$, we can replace EM with Viterbi updates by assigning inputs to their most likely Gaussian components (in their respective classes), and re-estimate parameters of a Gaussian component using only its members. In this way, the updates are greatly simplified. For example, for Gaussian means we have

$$\hat{\mu}_{y,k} = \alpha\sum_{i=1}^m \mathrm{I}(y_i = y, k_i = k)\mathbf{x}_i + (1-\alpha)\mu_{y,k}^{tr} \tag{5.8}$$

where α is a tradeoff parameter similar to λ.

It is worth noting that the EM (or Viterbi) algorithm can also be applied to fully unsupervised adaptation, where both y_i and k_i are missing for $\mathbf{x}_i$. In this case, we can simply replace $\delta_y(i)$ by a posterior probability $p(y_i = y|\mathbf{x}_i)$ in all update equations above. Unsupervised adaptation, however, can be problematic for classifying i.i.d. samples. This is because without any contextual constraints, an unadapted model is likely to make consistent errors, and fully unsupervised adaptation may amplify these errors rather than correcting them. For non-i.i.d. samples, however, we can impose high-level constraints, *e.g.* in the form of $p(y_{1:m})$, which are vital to the effectiveness of unsupervised adaptation. In continuous speech recognition, for example, $p(y_{1:m})$ is determined by a set of pronunciation models, which model the likelihood of a phoneme sequence in a word, and by a language model, which models the likelihood of a word sequence. In this case, the errors committed by the acoustic model may be corrected by the pronunciation models or the language model. In our classification experiments (Section 7.1.4), however, we are not coping with structured data, and we hence assume that $y_{1:m}$ are independent variables and are *known* in adaptation.

5.2 Links between SVMs and MLPs

The accuracy-regularization view of classification establishes a strong relationship between SVMs, MLPs, CRFs and MaxEnt models [143, 144]. We are particularly interested in the links between SVMs and MLPs, for a binary classification problem, the decision functions of SVMs and MLPs take on the same form:

$$f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b \tag{5.9}$$

where $\phi(\cdot)$ is a nonlinear transformation from the input space to a feature space in which linear classification takes place. Given this mapping, the objective of SVM training coincides with that of training the last layer of an MLP with weight decay [145], both of which can be expressed as

$$\min_{\mathbf{w},b} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} Q(f(\mathbf{x}_i), y_i) \tag{5.10}$$

where $\|\mathbf{w}\|^2$ is the squared ℓ_2 norm, $Q(\cdot)$ is a loss function, and λ is an accuracy-regularization tradeoff coefficient. Here f is represented by $(\mathbf{w}, b)$ as shown in Equation (5.9). There are several key differences between SVMs from MLPs: (1) the choice of the nonlinear transformation $\phi(\cdot)$; (2) how the parameters of $\phi(\cdot)$ are estimated; (3) the choice of the loss function $Q(\cdot)$; and (4) how the parameters of f (*i.e.*, $\mathbf{w}$ and b) are estimated. We discuss these aspects for SVMs and MLPs respectively in the following text.

For SVMs, $\phi(\cdot)$ is an input-to-feature space mapping that is implicitly determined by a reproducing kernel in the form of $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ [61, 78, 146]. The kernel function $k(\mathbf{x}, \mathbf{x}')$ measures certain "similarity" between two inputs. For example, the linear kernel has the form $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$, and the Gaussian kernel has the form $k(\mathbf{x}, \mathbf{x}') = -\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{\sigma^2}$. The parameters of the kernel function, such as σ^2 in the Gaussian kernel, are usually empirically chosen using cross-validation, though there are strategies for automatically optimizing kernel parameters [147] which are beyond the scope of this work.

In training a binary SVM, we typically assume a hinge loss function,

$$\xi_i \triangleq Q(f(\mathbf{x}_i), y_i) = |0, 1 - y_i f(\mathbf{x}_i)|_+. \tag{5.11}$$

where ξ_i are called *slack variables*. Since the hinge loss is not differentiable, gradient based optimization methods can not directly be applied. The training objective, however, can be formulated as a constrained convex optimization problem which has a globally optimal solution. Specifically,

$$\begin{aligned} \min_{\mathbf{w}, b, \xi_i} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b) + \xi_i - 1 \geq 0; \\ & \xi_i \geq 0, \end{aligned} \tag{5.12}$$

where $C = \dfrac{1}{\lambda m}$. Introducing the Lagrangian form, we arrive at the optimal solution as follows,

$$f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b = \langle \sum_{i=1}^{m} \alpha_i y_i \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle + b = \sum_{i=1}^{m} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \tag{5.13}$$

where $\mathbf{x}_i$ with nonzero α_i are called "support vectors" (SVs) and where the "kernel trick" is used so that $\phi(\cdot)$ is never explicitly evaluated [61, 78, 146]. The Lagrange multipliers α_i are obtained in the dual space by solving the following quadratic programming problem:

$$\begin{aligned} \max_{\alpha_i} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} \quad & \sum_{i=1}^{m} \alpha_i y_i = 0; \\ & 0 \leq \alpha_i \leq C \end{aligned} \tag{5.14}$$

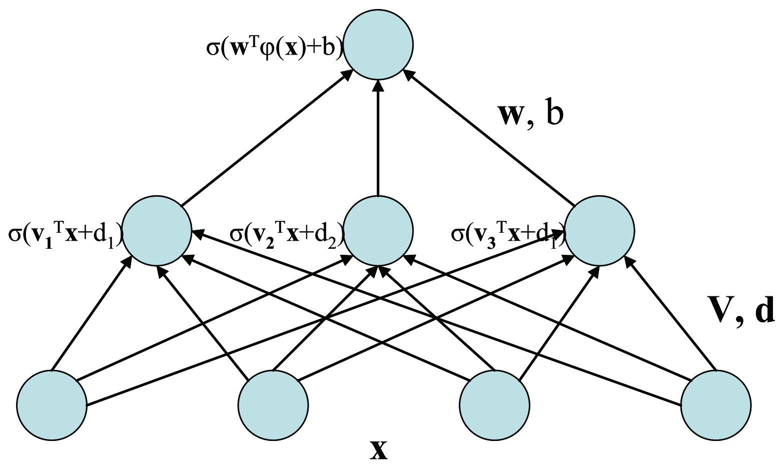


Figure 5.1: Multilayer perceptrons

Additionally, to handle multi-class classification, we can apply the "one-against-others" training scheme, *i.e.*, we learn a binary classifier, for each class, using all training samples labeled as either positive or negative w.r.t. this class. We adopt this scheme in our classification experiments since it is easy to implement and empirically works well [61]. But there are alternative approaches such as building pairwise classifiers [148] or using a training objective designed explicitly for multiclass classification [69, 149].

Secondly, we inspect these aspects in a two-layer MLP. For consistency with a binary SVM, we study a binary MLP which is depicted in Figure 5.1. In this example, we have D input units represented by a vector $\mathbf{x}$, N hidden units represented by a vector $\phi(\mathbf{x})$, and one output unit z indicating a binary decision. The weight vector and bias of the hidden-to-output layer are denoted by $\mathbf{w}$ and b respectively, and those of the input-to-hidden layer are denoted by $\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_N]^T$ and $\mathbf{d} = [d_1 \dots d_N]^T$. The *forward-propagation* of an MLP works as follows: the input vector goes through an affine transform followed by a non-linearity, producing a hidden node vector. Mathematically, $\phi_n(\mathbf{x}) = \sigma(\langle \mathbf{v}_n, \mathbf{x} \rangle + d_n)$, $n = 1..N$, where $\sigma(\cdot)$ is a sigmoid function. In the same way, the hidden-to-output layer produces a binary output $z = \sigma(f(\mathbf{x})) = \sigma(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b)$, where the sigmoid converts $f(\mathbf{x})$ to a valid posterior probability, *i.e.*, $z = p(y = +1|\mathbf{x}, f)$.

Under this setting, we see that the form of $\phi(\cdot)$ is fully determined once the number of hidden units is fixed. Note that there can be other choices of $\phi(\cdot)$ for general neural networks, such as the use of radius basis functions [145]. Moreover, the parameters in $\phi(\cdot)$, namely the input-to-hidden layer weights, can be optimized systematically using the *back-propagation* algorithm [145, 150].

In training an MLP, we minimize the conditional likelihood loss (or equivalently the logistic loss) $Q(f(\mathbf{x}_i), y_i) = p(y_i|\mathbf{x}_i, f)$ which is equivalent to minimizing the relative entropy between the true class membership distribution and the MLP output. To see this,

$$-\ln p(y_i|\mathbf{x}_i, f) = \mathrm{I}_{\{y_i=+1\}} \ln \frac{\mathrm{I}_{\{y_i=+1\}}}{p(y_i=+1|\mathbf{x}_i, f)} + \mathrm{I}_{\{y_i=-1\}} \ln \frac{\mathrm{I}_{\{y_i=-1\}}}{p(y_i=-1|\mathbf{x}_i, f)}, \tag{5.15}$$

It has been proved that by minimizing the relative entropy using a sufficient amount of training data, we can obtain an MLP whose output converges to the true posterior probability [145]. Notice that a sigmoid function is not convex, but its the logarithm is. Therefore, the training objective in Equation (5.10) is convex in the hidden-to-output layer parameters but not in the input-to-hidden layer parameters. Stochastic gradient descent or second order gradient methods [145, 150] are usually applied to find a local optimum.

The objective in Equation (5.10), in fact, can be readily extended to multi-class classification. In this case, the hidden-to-output weight vector is replaced by a weight matrix and the output sigmoid function is replace by a logistic function, *i.e.*, the k^{th} output is expressed as

$$z_k = \frac{\exp\{\mathbf{w}_k^T\mathbf{x} + b_k\}}{\sum_{k=1}^{K} \exp\{\mathbf{w}_k^T\mathbf{x} + b_k\}} \tag{5.16}$$

It again has the desired property that z_k converges to the true posterior probability if we minimize the relative entropy using a sufficiently large amount of training data.

As the community learns more about both MLPs and SVMs, their interpretation appears to be converging towards one idea. In fact, [143] showed that training an MLP with either weight decay or early stopping can be viewed as max-margin training, and [151] showed that the use of relative entropy as a loss function also has a max-margin interpretation.

As a side note, CRFs [152] and MaxEnt models [153, 154] has the same decision function form as Equation (5.9), and the same general training objective as Equation (5.10). In both cases, the choice of $\phi(\mathbf{x})$ is more flexible than that in SVMs or MLPs; it may essentially be any feature function, thereby facilitating the incorporation domain knowledge in the model.

Although the training algorithms for MLPs and SVMs are well studied, their respective adaptation algorithms are perhaps relatively under-investigated. In the next two sections, we present simple and principled adaptation algorithms derived from Chapter 2 for adapting SVMs and MLPs respectively. These algorithms, as we will see, provide a unified view of many existing SVM and MLP adaptation strategies.

5.3 SVM Adaptation

This section investigates SVM adaptation algorithms. We first review related work in the literature, and then present our algorithms derived from Chapter 4.

5.3.1 Related work

Most works on SVM adaptation in the literature follow an *incremental learning* or *sequential learning* paradigm (see [155] for early work on incremental learning). Incremental SVM learning was originally proposed to scale up inductive learning algorithms for very large datasets [132, 156, 157]. As discussed in the last section, *exact* SVM training requires to solve a quadratic programming problem in a number of coefficients equal to the number of training samples, thereby making large-scale learning problems difficult. Since only the support vectors (SVs) contribute to the decision boundary, training using these SVs would give exactly the same decision function as training on the whole data set. This makes SVM amenable to incremental learning, e.g. chunking [61], where the SVs of the previous subset are combined with a new subset in the next learning step.

The adaptation problem can be tackled in the same fashion – an adapted model can be learned using a combination of the SVs of the unadapted model and a subset of the adaptation data [130, 131]. In particular, [130] combined the old SVs with only mis-classified samples from the adaptation data; those correctly classified by the unadapted model are discarded since, to some extent, they are redundant given the SVs from the unadapted model. In Section 7.1.4, a classifier trained in this fashion will be referred to as a ”boosted” classifier since it strongly relates to the idea of boosting [79, 80]. In contrast, [131] took an exactly opposite strategy, which discarded the mis-classified adaptation samples while keeping the correctly-classified ones. The justification for this approach, however, was not clearly stated.

In this work, we would like to note that the original setting of incremental learning and the setting of adaptation are not entirely the same. The former aims to minimize the expected risk of the training distribution (which is presumably the same as the target distribution), while the latter aims to minimize the expected risk of the target distribution (which is different from the training distribution). The two data selection strategies [130] and [131] are more often used for incremental learning, but are not necessarily optimal for adaptation. In fact, there is no reason we should discard any adaptation data, because each sample is a true representative of the target distribution.

An improved incremental learning approach applicable to the problem of adaptation was introduced in [132]. In this work, the old SV set and the new data set are weighted differently in optimization. In the same spirit, [133] proposed an auxiliary data technique explicitly for scenarios where two data sources can have different distributions. This approach is the most similar to our work, as will be discussed shortly. A more general approach was proposed in our work [50], where each old SV was assigned a different weight, according to its likelihood with respect to the target distribution. We discuss this algorithm in the following subsection.

5.3.2 Error weighting – an empirical attempt

Specifically, we weight the slack variables to make an error on the training data less costly than one on the adaptation data. Again we only consider a binary classifier. We define SV^{tr} as the SVs obtained from the training data, and $(\mathbf{w}^{tr}, b^{tr})$ the resulting affine function parameters. Similarly, we define SV^{ts} as the SVs learned using only the adaptation data, and $(\mathbf{w}^{ts}, b^{ts})$ the resulting affine function parameters. Recall that the adaptation data is denoted as $\mathcal{D}_m^{ad}$ where m is the sample size. We then modify the objective function in Equation (5.12) to the following,

$$\begin{array}{ll} \min\limits_{\mathbf{w},b,\xi} & \frac{1}{2}\|\mathbf{w}\|^2 + C\Big(\sum\limits_{i=1}^{|SV^{tr}|} p_i\zeta_i + \sum\limits_{i=1}^{m}\xi_i\Big) \\ \text{subject to} & y_i(\langle\mathbf{w},\phi(\mathbf{x}_i)\rangle + b) + \zeta_i - 1 \geq 0; \;\; (\mathbf{x}_i, y_i) \in SV^{tr} \\ & y_i(\langle\mathbf{w},\phi(\mathbf{x}_i)\rangle + b) + \xi_i - 1 \geq 0; \;\; (\mathbf{x}_i, y_i) \in \mathcal{D}_m^{ad} \\ & \zeta_i \geq 0; \;\; \xi_i \geq 0 \end{array} \tag{5.17}$$

where p_i are weights for the slack variables of the old support vectors. In this way, we can adjust how important the role that the unadapted model plays in the adapted classifier. In an extreme case, where $p_i = 1$ for all i, the above objective is equivalent to training a SVM using all old SVs and all adaptation data. At the other extreme, where $p_i = 0$ for all i, the adaptation leads to a completely new SVM trained using only the adaptation data. Between these two extremes, we would like to weight each sample in SV^{tr} by how likely it is to be generated from the adaptation data distribution. For example, we can use

$$p_i = g(\langle \mathbf{w}^{ts}, \phi(\mathbf{x}_i) \rangle + b^{ts}), \tag{5.18}$$

where $(\mathbf{w}^{ts}, b^{ts})$ is the SVM trained on the adaptation data only, and $g(\cdot)$ is a monotonically increasing function converting a real number to a probability. For efficiency, we use an indicator function $g(x) = \mathrm{I}(x > d)$, where d is a regularization threshold. In such a setting, all SV^{tr} are selected when $d = -\infty$, while none are selected when $d = +\infty$. For comparison, in our experiments in Section 7.1.4 we also include the case where $d = 0$. In other words, we first retrain a rough classifier $(\mathbf{w}^{ts}, b^{ts})$ using only the adaptation data; then we use $(\mathbf{w}^{ts}, b^{ts})$ as a seed to select more samples (from SV^{tr}) to enhance this classifier. We will call this algorithm as "enhanced" in Section 7.1.4.

5.3.3 Regularized adaptation

In Chapter 4, inspired by the fidelity prior for log linear models, we proposed a regularized adaptation objective in Equation (4.29) repeated below for convenience. This leads to a simple yet principled approach to SVM adaptation. Here we let $\lambda_2 = 0$, meaning that b is estimated solely from the adaptation data. Moreover, we choose the value of $\lambda_1 = \lambda$ by cross-validation. What follows is the resulting adaptation objective.

$$\begin{aligned} \min_{\mathbf{w}, b, \xi_i} \quad & \frac{1}{2}\|\mathbf{w} - \mathbf{w}^{tr}\|^2 + C\sum_{i=1}^{m} \xi_i \\ \text{subject to} \quad & \xi_i \geq 1 - y_i(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b); \quad (\mathbf{x}_i, y_i) \in \mathcal{D}_m^{ad}; \\ & \xi_i \geq 0; \end{aligned} \tag{5.19}$$

where $\mathbf{w}^{tr}$ is the unadapted model. In contrast to SVM training, instead of minimizing $\|\mathbf{w}\|^2$, now we penalize the deviation from the unadapted model $\mathbf{w}^{tr}$. Note that there is a similar formulation referred to as "biased regularization" in [61] and used for incremental training of SVMs in [141].

Next, we show that the regularized adaptation objective in Equation (5.19) corresponds to max-margin training with a modified constraint. We first study Equation (5.19). We let $f^{tr}(\mathbf{x}_i) \triangleq$

$\langle \mathbf{w}^{tr}, \phi(\mathbf{x}_i)\rangle + b^{tr}$ denote the unadapted model, where $\mathbf{w}^{tr}$ and b^{tr} are presumed to be fixed during adaptation. Letting $\bar{\mathbf{w}} = \mathbf{w} - \mathbf{w}^{tr}$ and $\bar{b} = b - b^{tr}$, and plugging them into Equation (5.19), we have

$$\begin{aligned}
&\min_{\mathbf{w},b,\xi_i} && \frac{1}{2}\|\bar{\mathbf{w}}\|^2 + C\sum_{i=1}^{m}\xi_i \\
&\text{subject to} && \xi_i \geq 1 - y_i(\langle \bar{\mathbf{w}} + \mathbf{w}^{tr}, \phi(\mathbf{x}_i)\rangle + \bar{b} + b^{tr});\;\; \xi_i \geq 0 \\
&\Longrightarrow && \\
&\min_{\bar{\mathbf{w}},\bar{b},\xi_i} && \frac{1}{2}\|\bar{\mathbf{w}}\|^2 + C\sum_{i=1}^{m}\xi_i \\
&\text{subject to} && \xi_i \geq 1 - y_i\bar{f}(\mathbf{x}_i) - y_i f^{tr}(\mathbf{x}_i);\;\; \xi_i \geq 0.
\end{aligned} \tag{5.20}$$

where $\bar{f}(\mathbf{x}_i) = \langle \bar{\mathbf{w}}, \phi(\mathbf{x}_i)\rangle + \bar{b}$. We see that this is equivalent to the standard SVM training objective with a modified constraint. Similarly, we can derive the same constraints for Equation (5.27).

Figure 5.2 graphically illustrates how regularized adaptation works using an example of a linear SVM classifier on 2D input features. In the figure, (a) shows the adaptation data (depicted as solid circles and stars) and the decision function learned only using these samples, *i.e.*, $\langle \mathbf{w}^{ts}, \mathbf{x}\rangle + b^{ts}$; (b) shows the unadapted decision function $\langle \mathbf{w}^{tr}, \mathbf{x}\rangle + b^{tr}$, and those adaptation samples that have large margins w.r.t. to this decision boundary (depicted as hollow circles and stars) – little penalty should be put on these samples according to Equation (5.20); and (c) shows the adapted decision function obtained by training on the most relevant samples (depicted as solid circles and stars).

Before we present how to implement this adaptation algorithm, we would like to note that a number of existing SVM adaptation strategies including [130,131] can be unified by a generalization of the constraints in SVM training.

$$\begin{aligned}
&\min_{\mathbf{w},b} && \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\
&\text{subject to} && \xi_i \geq \psi(\mathbf{x}_i, y_i);\;\; (\mathbf{x}_i, y_i) \in \mathcal{D}_m^{ad} \\
& && \xi_i \geq 0;
\end{aligned} \tag{5.21}$$

where as usual $\mathcal{D}_m^{ad}$ denotes the adaptation set with m samples. Next we will illustrate that different forms of $\psi(\cdot,\cdot)$ yield different adaptation strategies.

- If we let $\psi(\mathbf{x}_i, y_i) = 1 - y_i f(\mathbf{x}_i)$, then Equation (5.21) is equivalent to Equation (5.10) with the hinge loss. The resulting adaptation algorithm only uses the adaptation data and entirely ignores information from the training data.

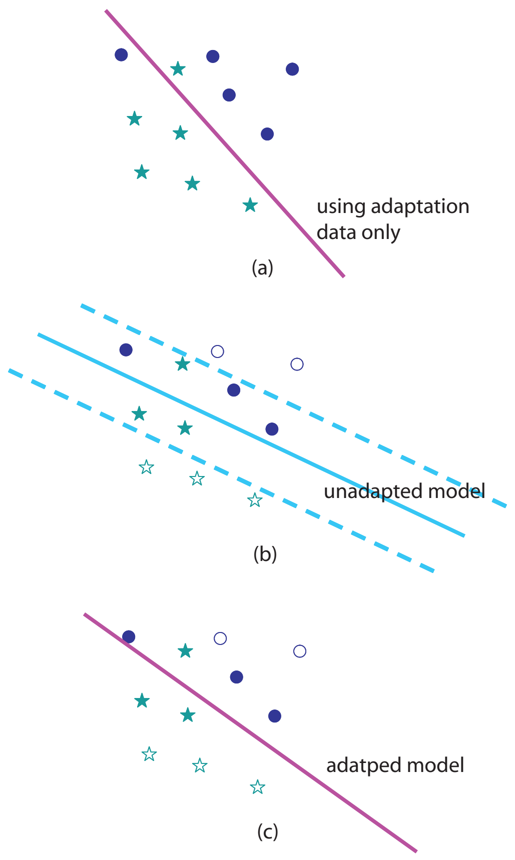


Figure 5.2: Regularized SVM adaptation. Circles and stars represent the adaptation data; the solid lines in (a), (b) and (c) respectively represent the entirely retrained model, the unadapted model, and the adapted model learned using Equation (5.20).

- If we let $\psi(\mathbf{x}_i, y_i) = \mathrm{I}(y_i f^{tr}(\mathbf{x}_i) < 0)(1 - y_i f(\mathbf{x}_i))$, where $\mathrm{I}(\cdot)$ is the indicator function, an adaptation sample is used only when it is misclassified by the unadapted model. If the objective minimizes the empirical risk on *both* old support vectors and the adaptation data, this method is equivalent to the method used in [130].

- Finally, our new approach combines the margin of the unadapted model and that of the model we are optimizing, creating a form of "soft boosting". Specifically, we let $\psi(\mathbf{x}_i, y_i) = 1 - y_i f(\mathbf{x}_i) - y_i f^{tr}(\mathbf{x}_i)$. Intuitively, if an adaptation sample has a large margin with respect to the decision boundary of the unadapted model, we decrease the importance of any margin error made by the adapted model in its total contribution to the loss. In other words, if $y_i f^{tr}(\mathbf{x}_i)$ is large and positive, then we penalize little on this sample even if $y_i f(\mathbf{x}_i) < 1$.

5.3.4 Algorithm derivation and implementation

This subsection discusses the derivation and implementation of the regularized adaptation algorithm proposed in Equation (5.19). Applying Lagrange multipliers $\alpha_i \geq 0$ (to enforce the margin constraints) and $\mu_i \geq 0$ (to enforce positivity of ξ_i), we have

$$J = \frac{1}{2}\|\mathbf{w} - \mathbf{w}^{tr}\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i\Big(y_i(\langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle + b) - 1 + \xi_i\Big) - \sum_{i=1}^{m}\mu_i\xi_i \tag{5.22}$$

Taking derivatives with respect to $\mathbf{w}$, b and ξ_i and setting them to zeros,

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{w}} = 0 \quad &\Longrightarrow \quad \mathbf{w} = \mathbf{w}^{tr} + \sum_{i=1}^{m}\alpha_i y_i \phi(\mathbf{x}_i) \\ \frac{\partial J}{\partial b} = 0 \quad &\Longrightarrow \quad \sum_{i=1}^{m}\alpha_i y_i = 0 \\ \frac{\partial J}{\partial \xi_i} = 0 \quad &\Longrightarrow \quad C - \alpha_i - \mu_i = 0 \end{aligned} \tag{5.23}$$

Using the “kernel trick” [61], we obtain the optimal decision function as

$$\begin{aligned} f(\mathbf{x}) &= \operatorname{sgn}\Big(\langle \mathbf{w}, \phi(\mathbf{x})\rangle + b\Big) \\ &= \operatorname{sgn}\Big(\sum_{i=1}^{|SV^{tr}|}\alpha_i^{tr}y_i^{tr}k(\mathbf{x}_i^{tr}, \mathbf{x}) + \sum_{i=1}^{m}\alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b\Big), \end{aligned} \tag{5.24}$$

where $(\mathbf{x}_i^{tr}, y_i^{tr})$ are support vectors from the unadapted model and α_i^{tr} are their corresponding weights which are fixed during adaptation. The Lagrangian multipliers α_i are to be optimized in the dual space using the adaptation data $\mathcal{D}_m^{ad}$. Plugging Equations (5.23) back into Equation (5.22), we arrive at an optimization problem in the dual space.

$$\begin{aligned} \max_{\alpha_i} \quad & \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{m}\alpha_i y_i \sum_{j=1}^{|SV^{tr}|}\alpha_j^{tr}y_j^{tr}k(\mathbf{x}_i, \mathbf{x}_j^{tr}) \\ \text{subject to} \quad & \sum_{i=1}^{m}\alpha_i y_i = 0; \\ & 0 \leq \alpha_i \leq C \end{aligned} \tag{5.25}$$

Fixing parameters from the unadapted model, and using the fact that $\sum_{i=1}^{m} \alpha_i y_i = 0$, the above is equivalent to

$$\max_{\alpha_i} \quad \sum_{i=1}^{m}(1 - y_i f^{tr}(\mathbf{x}_i))\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{5.26}$$

under the same constraints. We see that this differs from Equation (5.14) only in that α_i in the first term are weighted differently.

Alternatively, since the support vectors from the unadapted model are available at adaptation time, we can as well update their weights, *i.e.* α_j^{tr} in Equation (5.24). Note that we do this by minimizing the empirical risk on both the old support vectors and the adaptation data, leading to the following objective

$$\begin{aligned}
\max_{\alpha_i, \alpha_i^{tr}} \quad & \sum_{i=1}^{|SV^{tr}|} \alpha_i^{tr} + \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{|SV^{tr}|}\sum_{j=1}^{|SV^{tr}|} \alpha_i^{tr}\alpha_j^{tr} y_i^{tr} y_j^{tr} k(\mathbf{x}_i^{tr}, \mathbf{x}_j^{tr}) \\
& -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{m}\alpha_i y_i \sum_{j=1}^{|SV^{tr}|} \alpha_j^{tr} y_j^{tr} k(\mathbf{x}_i, \mathbf{x}_j^{tr}) \\
\text{subject to} \quad & \sum_{i=1}^{|SV^{tr}|} \alpha_i^{tr} y_i^{tr} = 0; \ \sum_{i=1}^{m} \alpha_i y_i = 0; \\
& 0 \le \alpha_j^{tr} \le C; \ 0 \le \alpha_i \le C
\end{aligned} \tag{5.27}$$

In our experiments, we call this the "extended regularized" algorithm. However, for SVMs using linear kernel, the old support vectors do not have to be saved — only the $\mathbf{w}$ and b are necessary to classification — and the "extended regularized" algorithm is not applicable in this case.

Regarding implementation, our optimization algorithm is developed based on SVMTorch [158], which uses a slightly modified version of the sequential minimal optimization (SMO) algorithm [159]. SMO is a conceptually simple algorithm with better scaling properties than a standard chunking algorithm. In the inner loop of the quadratic programming, SMO empirically chooses two α_i's at a time and optimizes them analytically. For reference Appendix B.2 gives the updating formula for our regularized adaptation algorithm which is derived based the SMO algorithm [159].

Although SMO greatly expedites the optimization process, the algorithm can still converge slowly especially when using nonlinear kernels on very large datasets (*e.g*, over 20K samples). Moreover, it is sometimes expensive to store and transfer a SVM classifier as the number of SVs in general scales up with the number of training samples [61]. Since the regularized adaptation algorithm and many other algorithms require the evaluation of $f^{tr}(\mathbf{x}_i)$ at every adaptation sample i, the time and space efficiency can be adversely affected by the scale of the unadapted model.

5.4 MLP Adaptation

This section discusses adaptation algorithms for MLP classifiers. Analogous to the last section, we first review related work and then present our proposed regularized adaptation algorithm.

5.4.1 Related work

The problem of MLP adaptation has been investigated in the machine learning community in the context of multi-task learning [26, 28, 31], and in the speech community for adapting hybrid HMM-MLP systems [127–129, 160]. A common adaptation strategy is either retraining part of the original network or adding augmentative layers. The network parameters to be retrained or added can be estimated via back-propagation [145, 150]. First, there is RSI (*retrained speaker-independent*), discussed in [128]. This approach starts from the unadapted (speaker-independent) model and retrains the entire network, while early stopping [143] can be applied to avoid overfitting. Next, there is RLL (*retrained last layer*) [26, 31, 127] which starts from the unadapted model and retrains only the hidden-to-output layer of the MLP. The intuition is that the input-to-hidden layer provides an "internal representation" which is common to all tasks, while the hidden-to-output layer provides a task-dependent decision function constructed on the basis of this internal representation [26]. The method in [127] also enables the selection of the most active hidden neurons, and it only adapts the last-layer weights associated with these neurons. Finally, another popular approach LIN (*linear input network*) [128, 129] augments the unadapted network with an additional linear transformation input layer. It is this layer that is trained using the adaptation data; the original network stays unchanged. In this case, the number of free parameters is often further reduced using parameter tying. For example, in vowel classification or speech recognition, the input feature vector is often a concatenation of feature vectors from a number of consecutive frames. The transformation matrix, therefore, is often constrained to be a block diagonal matrix with identical block-wise sub-matrices.

In many real-world tasks, these methods have shown significant performance improvements over merely using the unadapted model. None of them, however, has worked universally the best in all adaptation scenarios. One reason is that the number of free parameters to estimate depends on the MLP architecture. For example, in object recognition tasks, the input layer often has a very high dimension D that equals the number of pixels in a raw image. In such a case, the LIN algorithm that directly estimates a gigantic $D \times D$ transformation matrix is likely to fail. Although parameter tying might mitigate the overfitting problem, it is not clear how to perform typing on an input image. The second reason is that the amount of labeled adaptation data may vary from task to task, but the degrees of freedom in the above algorithms are more or less fixed, which can easily cause overfitting or underfitting.

5.4.2 Regularized adaptation

In this work, we apply a regularized adaptation algorithm which is amenable to changes in MLP architecture and in the amount of adaptation data available. The adaptation objective is derived from Equation (4.29), where we use the relative entropy as the $Q(\cdot)$ function and where we let $\lambda_2 = \lambda$ similar to the case of SVMs.

In fact, we can extend this algorithm to a multi-class, two-layer MLP where we regularize the weight matrix in each layer of the network. We regularize the input-to-hidden layers only because we have found it to be practically advantageous as will be shown in Section 7.1.4 — this regularizer is not derived from our fidelity prior. Here we use the same notation as was defined in Section 5.2. Since we are considering K classes, the hidden-to-output layer now consists of K weight vectors and biases, forming a weight matrix $\mathbf{W}$ and a bias vector $\mathbf{b}$.[1] Similarly, the input-to-hidden layer weight matrix and bias vector are denoted by $\mathbf{V}$ and $\mathbf{d}$ respectively. We also introduce a set of variables regarding sample i in Table 5.1

Given the above notation, the adaptation objective is expressed as

$$\min_{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{d}} \quad \frac{\lambda^{h2o}}{2}\|\mathbf{W} - \mathbf{W}^{tr}\|^2 + \frac{\lambda^{i2h}}{2}\sum \|\mathbf{V} - \mathbf{V}^{tr}\|^2 + \frac{1}{m}\sum_{i=1}^{m}\left(\sum_{k=1}^{K} t_{i,k}\log\frac{t_{i,k}}{z_{i,k}}\right) \tag{5.28}$$

where we use separate tradeoff coefficients λ^{h2o} and λ^{i2h} for regularizing the input-to-hidden and hidden-to-output layers respectively, and where $\|A\|^2 = tr(AA^T)$. Due to mathematical tractability

[1] In fact, we can incorporate the bias into the weight vector if adding "1" as another dimension in the vector $\phi(\mathbf{x})$. In this work, we use explicit bias terms for consistency with the SVM notation.

Table 5.1: Notation of a two-layer MLP

$\mathbf{x}_i$:	input vector representing D input units;
$\mathbf{c}_i$:	$\mathbf{c}_i = \mathbf{V}\mathbf{x}_i + \mathbf{d}$; *i.e.*, $c_{i,n} = \langle \mathbf{v}_n, \mathbf{x}_i \rangle + d_n$, $n = 1..N$;
ϕ_i:	hidden vector representing N units, where $\phi_{i,n} = \sigma(c_{i,n})$, $n = 1..N$;
$\mathbf{a}_i$:	$\mathbf{a}_i = \mathbf{W}\phi_i + \mathbf{b}$; *i.e.*, $a_{i,k} = \langle \mathbf{w_k}, \phi_\mathbf{i} \rangle + b_k$, $k = 1..K$
$\mathbf{z}_i$:	output vector representing K output units, $z_{i,k} = \dfrac{\exp\{-a_{i,k}\}}{\sum_{j=1}^{K} \exp\{-a_{i,j}\}}$, $k = 1..K$;
$\mathbf{t}_i$:	target label vector, where $t_{i,k} = \mathrm{I}(k = y_i)$

of the ℓ_2-norm, the above adaptation objective can be easily optimized using the back-propagation algorithm [145, 150] (stochastic gradient descent or second-order gradient methods), which is typically much faster than the quadratic-programming-like procedures needed for SVM optimization in terms of total computation time.

Many adaptation algorithms introduced in the past can fit in the regularized adaptation framework. First, the RSI algorithm, in which both layers are re-estimated, is akin to $\lambda^{h2o} = \lambda^{i2h} = 0$. Next, the RLL algorithm in which only the last layer is re-estimated is akin to $\lambda^{h2o} = 0$ and $\lambda^{i2h} \to \infty$. Finally, since a cascade of linear transforms is a linear transform, the method of LIN is akin to $\lambda^{h2o} \to \infty$ and $\lambda^{i2h} = 0$. We also define in this work a new method, entitled RFL (*retrained first layer*) which also corresponds to $\lambda^{h2o} \to \infty$ and $\lambda^{i2h} = 0$. Of course, all of these methods are generalized by varying the λ^{h2o} and λ^{i2h} tradeoff coefficients.

5.4.3 Algorithm derivation and implementation

This subsection discusses the derivation and implementation of the regularized adaptation algorithm in Equation (5.28). For convenience we define the last term in the objective as

$$J_i \triangleq \sum_{k=1}^{K} t_{i,k} \log \frac{t_{i,k}}{z_{i,k}} \tag{5.29}$$

Now we apply the back-propagation algorithm. The first step is to compute the first-order derivatives w.r.t. the model parameters to be adapted. Taking derivatives of J_i with respect to the hidden-to-output layer parameters $\mathbf{w}_k$ and b_k, we have

$$\begin{aligned}
\frac{\partial J_i}{\partial \mathbf{w}_k} &= \frac{\partial J_i}{\partial a_{i,k}} \cdot \phi_i \\
&= \sum_{j=1}^{K} \frac{\partial J_i}{\partial z_{i,j}} \cdot \frac{\partial z_{i,j}}{\partial a_{i,k}} \cdot \phi_i \\
&= \sum_{j=1}^{K} (-\frac{t_{i,j}}{z_{i,j}}) \cdot (z_{i,j}\delta_{j,k} - z_{i,j}z_{i,k}) \\
&= (z_{i,k} - t_{i,k}) \cdot \phi_i \\
\frac{\partial J_i}{\partial b_k} &= z_{i,k} - t_{i,k}
\end{aligned} \tag{5.30}$$

Next, taking derivatives of J_i w.r.t. the input-to-hidden layer parameters $\mathbf{v}_n$ and d_n gives

$$\begin{aligned}
\frac{\partial J_i}{\partial \mathbf{v}_n} &= \frac{\partial J_i}{\partial c_{i,n}} \cdot \mathbf{x}_i \\
&= \sum_{k=1}^{K} \frac{\partial J_i}{\partial a_{i,k}} \cdot \frac{\partial a_{i,k}}{\partial c_{i,n}} \cdot \mathbf{x}_i \\
&= \phi_{i,n}(1 - \phi_{i,n}) \sum_{k=1}^{K} (z_{i,k} - t_{i,k}) w_{k,n} \cdot \mathbf{x}_i \\
\frac{\partial J_i}{\partial d_n} &= \phi_{i,n}(1 - \phi_{i,n}) \sum_{k=1}^{K} (z_{i,k} - t_{i,k}) w_{k,n}
\end{aligned} \tag{5.31}$$

On the other hand, it is straightforward to derive the derivative of $\|\mathbf{w} - \mathbf{w}^{tr}\|^2$. Given these derivatives, the second step is to update the model parameters using stochastic gradient descent. In other words, we apply gradient descent using *online* processing where the parameters are updated every sample, in contrast to *batch* processing where each update is averaged over all samples. Online processing often yields better performance since the parameters are updated much more frequently than in the batch mode. Additionally, *bunch* processing is a compromise between the two, in which each update is averaged over a "bunch" of samples. The inner loop of the gradient descent algorithm for online processing is written as follows. For $i = 1..m$,

$$\mathbf{w}_k = \mathbf{w}_k - \eta\Big((z_{i,k} - t_{i,k}) \cdot \phi_i + \lambda^{h2o}(\mathbf{w}_k - \mathbf{w}_k^{tr})\Big) \tag{5.32}$$

$$b_k = b_k - \eta(z_{i,k} - t_{i,k}) \tag{5.33}$$

$$\mathbf{v}_n = \mathbf{v}_n - \eta\Big(\phi_{i,n}(1 - \phi_{i,n}) \sum_k (z_{i,k} - t_{i,k}) w_{k,n} \cdot \mathbf{x}_i + \lambda^{i2h}(\mathbf{v}_n - \mathbf{v}_n^{tr})\Big) \tag{5.34}$$

$$d_n = d_n - \eta\Big(\phi_{i,n}(1 - \phi_{i,n}) \sum_k (z_{i,k} - t_{i,k}) w_{k,n}\Big) \tag{5.35}$$

where η is the learning rate. We will give the update formula for a general L-layer MLP in Appendix B.3.

Given a two-layer MLP with D input units, N hidden units and K output units, the space complexity of the MLP is $O((D+K)N)$. The time complexity of the adaptation *inner* loop is $O((D+K)Nm)$, where m (the number of adaptation samples), while the number of outer loops (or learning epoches) may vary. In general, MLPs are more efficient than SVMs on large datasets, while SVMs perform better on sparse data.

5.5 Relation to Inverse Optimization Problem

Before we move on to the experiment section, we discuss the relationship between adaptation and *inverse combinatorial optimization* [161–163]. We use the shortest path problem as an example of combinatorial optimization problems. Assume that we have a graph in which nodes represent locations and edges represent roads connecting locations. Further assume that there is a cost associated with each road, which depends on length, road condition, traffic condition and etc. The shortest path problem is to find a path between two locations with the minimum sum of costs. Formally, considering a directed graph, we let $(i,j) \in E$ denote a directed edge from i to j, and let c_{ij} denote the cost associated with this edge. We further define a set of binary variables x_{ij} indicating whether the edge from i to j is selected. The problem of finding the shortest path from s to t is equivalent to finding x_{ij} such that $\sum c_{ij}x_{ij}$ is minimized under the constraint that x_{ij} constitute a valid path from s to t. This can be formulated as a linear programming problem [164]:

$$\begin{aligned} \min_{x_{ij}} \quad & \sum_{(i,j)\in E} c_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \begin{cases} 1 & i = s \\ 0 & i \neq s \text{ and } i \neq t \\ -1 & i = t \end{cases} \\ & 0 \leq x_{ij} \leq 1 \; \forall (i,j) \in E \end{aligned} \tag{5.36}$$

If the graph does not have any negative cost cycle, the above linear programming problem has an 0-1 optimal solution [164]. More generally, a family of optimization problems (including the shortest path problem) can be formulated as

$$\min_{\mathbf{x}\in\mathcal{A}} g(\mathbf{c}^T\mathbf{x}) \tag{5.37}$$

where A is a set of feasible solutions.

The inverse optimization problem is such that for a feasible solution to the original optimization problem, we want to discover under what cost vectors will this feasible solution become an optimal solution. The inverse problem has a number of variations [162–164], one of which is to find a new cost vector $\mathbf{c}$, with minimal deviation from some known cost vector $\mathbf{c}_0$, such that a given solution $\mathbf{x}_0$ is an optimal solution to (5.37), *i.e.*,

$$\begin{aligned} \min_{c} \quad & \|\mathbf{c}-\mathbf{c}_0\|_p \\ \text{s.t.} \quad & g(\mathbf{c},\mathbf{x}_0) = \min_{\mathbf{x}\in\mathcal{A}} g(\mathbf{c},\mathbf{x}) \end{aligned} \tag{5.38}$$

It has been shown by [163, 164] if (5.37) is a linear programming problem, *e.g.* the shortest path problem, then its inverse problem under norm $p = 1$ and $p = \infty$ is also a linear programming problem.

Slightly twisting the roles of these variables, we see that the inverse optimization problem closely relates to regularized adaptation. We replace $\mathbf{v}$ with class labels $y_{1:m}$ in a classification problem, replace $\mathbf{c}$ with affine parameters $(\mathbf{w}, b)$, and let $g(\cdot) = \sum_{i=1}^{m} Q(f(\mathbf{x}_i), y_i)$, where $f(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i + b$. In adaptation, we are given $\mathbf{c}_0 = (\mathbf{w}^{tr}, b^{tr})$ which is known a-priori, and we aim to estimate new parameters $(\mathbf{w}, b)$, with minimal deviation from $(\mathbf{w}^{tr}, b^{tr})$, such that the true class labels $\mathbf{v}_0 = y_{1:m}$ becomes an optimal solution w.r.t. $g(\cdot)$. Formally,

$$\begin{aligned} \min_{\mathbf{w}} \quad & \|\mathbf{w}-\mathbf{w}^{tr}\|^2 \\ \text{s.t.} \quad & \sum_{i=1}^{m} Q(f(\mathbf{x}_i), y_i) = \min_{y'_{1:m}\in\mathcal{Y}^m} \sum_{i=1}^{m} Q(f(\mathbf{x}_i), y'_i); \end{aligned} \tag{5.39}$$

where $(\mathbf{x}_i, y_i)$, $i = 1..m$, are samples from the adaptation set $\mathcal{D}_m^{ad}$.

We can prove that for binary classification and for certain loss functions, a decision function f satisfying the constraint in (5.39) has zero classification error on the adaptation data. To see this, since $(\mathbf{x}_i, y_i)$ are i.i.d., the constraint is equivalent to

$$Q(f(\mathbf{x}_i), y_i) = \min_{y'_i=\pm 1} Q(f(\mathbf{x}_i), y'_i), \quad i = 1..m \tag{5.40}$$

We are especially interested in the hinge loss and the log loss, using which the above constraint is equivalent to

$$y_i f(\mathbf{x}_i) = \max_{y'_i=\pm 1} y'_i f(\mathbf{x}_i), \quad i = 1..m \tag{5.41}$$

Since $y_i \in \{\pm 1\}$, it must be that any decision function f satisfying Equation (5.41) satisfies $y_i f(\mathbf{x}_i) > 0$ for all i, which implies zero classification error. On the other hand, it is easy to see that a decision function with zero classification error on the adaptation data must satisfy the constraint in (5.39). This suggests that if $\mathbf{w}^{tr}$ already achieves zero classification error on $\mathcal{D}_m^{ad}$, then $\mathbf{w}^{tr}$ itself is the optimal solution to the inverse optimization problem in Equation (5.39), which is intuitively correct.

In practice, (5.40) is often too strict a constraint to allow any feasible solution. For example, when $\mathcal{D}_m^{ad}$ is non-separable, there is no decision function that satisfies (5.41). We therefore would like to relax the constraint a bit so that the problem generalizes to non-separable cases. We introduce a slack variable $\xi_i > 0$ to each of the constraints in (5.40). This slack variable represents how much the loss of the correct label exceeds the minimal loss of all possible labels, and we certainly want these slack variables to be as small as possible. The inverse optimization problem hence becomes

$$\begin{aligned} \min_{\mathbf{w}} \quad & \tfrac{\lambda}{2}\|\mathbf{w} - \mathbf{w}^{tr}\|^2 + \sum_{i=1}^{m} \xi_i \\ \text{s.t.} \quad & Q(f(\mathbf{x}_i), y_i) = \min_{y_i' \in \mathcal{Y}} Q(f(\mathbf{x}_i), y_i') + \xi_i; \\ & \xi_i \geq 0 \end{aligned} \tag{5.42}$$

This leads to an adaptation objective for potential applications such as parsing and structure learning of graphical models. A large-margin training objective with constraints in a similar fashion has been proposed in [69].

5.6 Adaptation Experiments

This section presents two sets of adaptation experiments, one on a vowel classification task and the other on an object recognition task. The general experimental paradigm is as follows: for each task, we learn the best unadapted model on the training set, and then perform adaptation and evaluation via cross-validation (CV) on the test set. In other words, we divide the test set into n folds, we repeatedly choose one fold as the adaptation set and the remaining folds as the test set.

The first task involves a reasonably large training set of about 300K samples. As we found that training a SVM on such a dataset was prohibitive, we used sub-sampled data to learn the unadapted model, and we will focus more attention on the use of MLPs in this task. The second task involves a rather sparse training set of 2700 image samples with high-dimensional features. As we will see, an unadapted SVM is advantageous over an unadapted MLP in this case. Since the performances of the unadapted SVM and MLP are different, it is to some extent unfair to compare the performances of their respective adapted models. Our goal here is to compare various adaptation algorithms applied to the same unadapted classifier.

5.6.1 Frame-level vowel classification

We have briefly introduced the Vocal Joystick in Chapter 1, and we will elaborate on this topic in Chapter 6. Along with the development of the VJ system, we have collected a large corpus of vowel utterances which can be potentially used in pattern recognition and machine learning research [55]. This section discusses the application of our proposed regularized adaptation algorithms to a vowel classification task based on this corpus.

The VJ-vowel corpus

The Vocal Joystick (VJ) [36] is a human-computer interface system that enables individuals with motor impairments to control computing devices (mice pointers, etc.) with continuous aspects of their voice (pitch, vowel quality). Within this project, we needed a *frame-level* vowel classifier. To this end, we initially used speech data from the TIMIT speech corpus [165] to train such a classifier. However, we empirical found that using TIMIT is not best-suited for training the VJ vowel classifier. This is because the vowels used by the VJ often need to be pronounced in a drawn-out manner, while the vowels in TIMIT are usually short and are often spoken with much co-articulation present. Therefore, we made a large data collection effort in a controlled environment that yields a new vowel corpus that was representative of the utterances a user of the VJ-system would use. The result of the data collection effort is a vowel corpus of approximately 11 hours of recorded data comprised of approximately 23,500 sound files of monophthongs and vowel combinations (e.g. diphthongs) with variations in duration, pitch and intensity. A full description of this corpus and the data collection methodologies can be found in [55].

To test our machine learning algorithms, we extracted from the VJ corpus a dataset of 8 monophthong vowels: /æ/, /a/, /ɑ/, /o/, /u/, /ɨ/, /i/, and /e/. All speakers articulated each vowel with all

Table 5.2: Amounts of training, test and development data in VJ-Vowel

	# speakers	# samples in total	non-silent audio
Training set	21	420K	1.16 hrs
Test set	10	200K	0.56 hrs
Development set	4	80K	0.22 hrs

combinations of the following: (a) long/short (in duration); (b) falling/level/rising (in pitch); (c) loud/normal/quiet (in intensity). In other words, there are $2 \times 3 \times 3 = 18$ utterances for each vowel class and each speaker. We allocated 21 speakers to the training set (for training the unadapted model), 10 speakers to the test set (for CV-style adaptation and evaluation) and 4 speakers to the development set (for parameter tuning). In this thesis, we use exactly the *same* test set for MLP, SVM and GMM adaptation experiments so that the results are comparable. Unlike earlier experiments [48,49], the audio files in this test set were recorded *without* phonetician's supervision, which we think is a more realistic scenario in real-world applications. The training and development sets, however, were collected with phonetician's supervision. The number of samples in each set is listed in Table 5.2. For a particular speaker in the development set and the test set, we performed 6-fold adaptation-evaluation experiments (same as cross-validation) with different amounts of adaptation data. Specifically, experiment VJ-A, VJ-B and VJ-C used approximately 1K, 2K and 3K samples per speaker for adaptation respectively, while all experiments used the same 17K samples per speaker for evaluation. We repeated this for all 6 folds and computed the average error rate for each speaker. We then computed the mean and standard deviations of these averaged error rates over 10 speakers. In this regard, the standard deviation reflects the variation in adaptation performance across different speakers (which is typically rather high). To test the significance of the mean error rates, we also conducted *difference of proportion* significance test.

Regarding input features, we created a frame every 10ms, each with a length of 25ms. We then extracted standard *mel frequency cepstral coefficients* (MFCCs) plus their deltas each frame, producing a 26-dimensional vector. Moreover, an input vector $\mathbf{x}_i$ is formed by concatenating

Table 5.3: Dev-set error rates of **unadapted** 8-class MLPs with different numbers of hidden units and window sizes. The highlighted entries include the best error rate and those **not** significantly different from the best at the $p < 0.002$ level.

hidden/win	1	3	5	7	9
25	29.92	27.20	26.03	**25.85**	25.92
50	28.07	27.30	**25.75**	**25.27**	26.07
100	27.83	26.80	26.87	26.30	26.37
200	27.80	26.53	26.4	26.03	25.97
400	27.37	26.23	25.97	25.90	25.83

MFCCs+deltas from a consecutive W frames centered at i, where W is the window size.

MLP experiments

First, we trained an unadapted two-layer MLP with different configurations. We varied the number of window size W on one hand, and the number of hidden nodes N on the other. For each setting, we also tuned the regularization coefficients λ^{h2o} and λ^{i2h}, and found that the impact of regularization was negligible in this case (probably because the training set is large). Note that all above parameters were tuned using the development set. The results are reported in Table 5.3, which shows that the best result comes from using a window size of 7 and using 50 hidden units. When applied to the test set, this configuration obtained an error rate of 33.14%. This error rate is quite high considering there are only 8 vowel classes. By listening to the waveforms, we found that although most of the vowel utterances from the same speaker were distinguishable from one another, it was fairly common that a vowel articulated by one speaker sounded similar to a different vowel articulated by another speaker, especially in the case of /a/ and /ɑ/. In other words, the Bayes error of a speaker-independent classifier tends to be high on this dataset.

Next, we conducted adaptation experiments based on the best vowel classifier we obtained. and compared the following algorithms: (1) **unadapted**; (2) **retrained** with weight decay using only the adaptation data; (3) **LIN** (linear input network) [128]; (4) **RSI** (retrained speaker independent)

Table 5.4: Test-set error rates (means and standard deviations over 10 speakers) of experiment VJ-A, VJ-B and VJ-C using 8-class MLPs, where the amounts of adaptation data were balanced across classes; The highlighted entries include the best error rate and those **not** significantly different from the best at the $p < 0.001$ level.

# samples per speaker	1K	2K	3K
Unadapted	33.14 $\pm$ 9.38	33.14 $\pm$ 9.38	33.14$\pm$ 9.38
Retrained	23.80 $\pm$ 5.17 (λ^{h2o}=10^{-3} λ^{i2h}=0)	17.94 $\pm$ 4.80 (λ^{h2o}=10^{-4} λ^{i2h}=0)	14.31 $\pm$ 3.97 (λ^{h2o}=0 λ^{i2h}=0)
LIN	21.87 $\pm$ 5.02	18.28 $\pm$ 4.89	16.56 $\pm$ 4.05
RSI (λ^{h2o}=0, λ^{i2h}=0)	18.97 $\pm$ 5.09	15.70 $\pm$ 4.25	12.90 $\pm$ 3.64
RLL (λ^{h2o}=0, λ^{i2h}=∞)	21.84 $\pm$ 5.74	18.40 $\pm$ 4.65	15.83 $\pm$ 3.77
RFL (λ^{h2o}=∞, λ^{i2h}=0)	**18.64** $\pm$ 5.37	15.20 $\pm$ 4.23	**12.18** $\pm$ 3.51
Regularized	**18.64** $\pm$ 5.37 (λ^{h2o}=∞ λ^{i2h}=0)	**15.20** $\pm$ 4.27 (λ^{h2o}=∞ λ^{i2h}=10^{-2})	**12.30** $\pm$ 3.68 (λ^{h2o}=∞ λ^{i2h}=10^{-4})

[128]; (5) **RLL** (retrained last layer) [127]; (6) **RFL** (retrained first layer), which we introduce in this work since it is a natural instantiation of regularized adaptation; and (7) **regularized** as presented in Equation (5.28). For (2) and (7), we first chose the best λ^{h2o} and λ^{i2h} values on the development set, and then applied them to the test set. In all cases, we computed the average error rates over these 6 adaptation/evaluation folds. We then repeated this for all 10 test speakers, and the final results are means and standard deviations of these error rates over 10 speakers.

Table 5.4 shows results for experiment VJ-A, VJ-B and VJ-C which used different amounts of adaptation data, *i.e.* 1K, 2K and 3K samples per speaker respectively (corresponding to 10s, 20s and 30s for all 8 vowels spoken by each speaker). The amounts of adaptation data were balanced across all 8 classes. As shown in the table, all adaptation algorithms outperformed the unadapted model, while their performance kept improving as more adaptation data became available. Overall, ”regularized” performed the best in terms of mean error rate, and RFL worked almost as well. Note that the high standard deviations are largely due to the variation in unadapted model performance

Table 5.5: Test-set error rates (means and standard deviations over 10 speakers), with different number of classes of adaptation data available (unbalanced classes), and with ~ 350 samples per class (3K in total). Highlighted entries include the best error rate and those **not** significantly different from the best at the $p < 0.001$ level.

# vowel classes in $\mathcal{D}_m^{ad}$	1	2	3	4
Unadapted	**33.14** ± 9.38	**33.14** ± 9.38	**33.14** ± 9.38	33.14 ± 9.38
LIN	41.01 ± 4.25	41.30 ± 6.36	39.46 ± 5.94	35.35 ± 6.80
RSI	37.42 ± 5.20	40.78 ± 4.43	40.43 ± 4.78	36.76 ± 5.57
RLL	38.24 ± 5.63	43.63 ± 5.40	44.62 ± 5.31	41.91 ± 6.32
RFL	34.89 ± 5.52	34.68 ± 4.30	34.67 ± 5.34	27.23 ± 6.15
Regularized	**33.14** ± 9.38 $\lambda^{h2o} = \infty$ $\lambda^{i2h} = \infty$	**33.14** ± 9.38 $\lambda^{h2o} = \infty$ $\lambda^{i2h} = \infty$	**33.14** ± 9.38 $\lambda^{h2o} = \infty$ $\lambda^{i2h} = \infty$	**25.44** ± 5.95 $\lambda^{h2o} = \infty$ $\lambda^{i2h} = 10^{-2}$
# vowel classes in $\mathcal{D}_m^{ad}$	5	6	7	8
Unadapted	33.14 ± 9.38	33.14 ± 9.38	33.14 ± 9.38	33.14 ± 9.38
LIN	27.82 ± 6.76	27.19 ± 4.40	21.29 ± 4.50	16.56 ± 4.05
RSI	29.80 ± 8.08	26.86 ± 5.41	17.44 ± 5.28	12.90 ± 3.64
RLL	35.27 ± 8.72	31.41 ± 5.10	22.55 ± 5.10	15.83 ± 3.77
RFL	21.85 ± 7.46	21.14 ± 5.46	13.73 ± 4.22	**12.18** ± 3.51
Regularized	**20.32** ± 6.28 $\lambda^{h2o} = \infty$ $\lambda^{i2h} = 10^{-2}$	**20.23** ± 5.29 $\lambda^{h2o} = \infty$ $\lambda^{i2h} = 10^{-2}$	**13.07** ± 4.20 $\lambda^{h2o} = \infty$ $\lambda^{i2h} = 10^{-4}$	**12.30** ± 3.68 $\lambda^{h2o} = \infty$ $\lambda^{i2h} = 10^{-4}$

when tested on different speakers (as indicated by the standard deviation of the unadapted model). Furthermore, for the Vocal Joystick application, we also trained a 4-class MLP. The unadapted model had an error rate of $8.11 \pm 2.37\%$. Regularized adaptation obtained $3.06 \pm 3.73\%$, $2.21 \pm 2.25\%$ and $1.98 \pm 2.02\%$ using 1K, 2K and 3K adaptation samples per speaker respectively.

An alternative approach to measure adaptation performance is to vary the fraction of the number of vowel classes available to adapt on (e.g., it may be desirable to adapt using data from only samples from 3 of those 8 vowel classes). This approach is more akin to the situation in ASR, where limited adaptation data almost assuredly means that certain categories (phones or words) are not available. Table 5.5 shows results when the amounts of adaptation data were unbalanced across the 8 classes. Column i means a system was adapted only with i out of the 8 possible number of classes. We experimented with different random orders that the vowel classes became available for adaptation and observed similar patterns in results, and here we only report the results for one such order. As the table shows, when the number of vowel classes available was no larger than 3, the best strategy was to use the unadapted model; when this number was above 3, "regularized" performed significantly better than any of the other algorithms up to all 8 vowel classes at which point "regularized" and RFL performed equally well. It is also interesting to notice that the best configuration for the regularized algorithm was almost always $\lambda^{h2o} = \infty$ and $\lambda^{i2h} \approx 0$, meaning that the last layer of the MLP is fixed and the first layer had moderate or no regularization.

SVM experiments

We first attempted to train an unadapted SVM using the entire training set, but found it unlikely to finish within weeks. We then used 80K samples randomly selected from the training set, and trained an unadapted SVM with Gaussian kernel. We empirically chose $std = 10$ using the development set and fixed this parameter in adaptation. In a similar way, we chose the regularization coefficient $C = 100$. The unadapted model had an error rate of 38.21%.

We then applied different SVM adaptation algorithms to the unadapted model. Specifically, we tried (1) **unadapted**; (2) **retrained** which trains a new SVM using the adaptation data only; (3) **boosted** [130] which combines the misclassified adaptation samples with the old SVs in training an adapted classifier; (4) **enhanced** which corresponds to the algorithm we proposed in Section 5.3.2; (5) **regularized** which follows Equation (5.19); and finally (6) **extended regularized** which follows Equation (5.27). As shown in Table 5.6, "retrained", "enhanced" and "regularized" worked very well in general, while "regularized" had a non-trivial gain over others when the adaptation data was very limited. Furthermore, we see that "regularized" performed better than "extended regularized."

Table 5.6: Test-set error rates (means and standard deviations over 10 speakers) of experiment VJ-A, VJ-B and VJ-C using SVMs with a Gaussian kernel (std=10). The tradeoff coefficient is $C = 100$ in all cases. The highlighted entries include the best error rate and those **not** significantly different from the best at the $p < 0.001$ level.

# adaptation samples per speaker	1K	2K	3K
Unadapted	38.21 ± 11.41	38.21 ± 11.41	38.21± 11.41
Retrained	24.70 ± 4.33	**18.94** ± 3.56	**14.00**± 3.32
Boosted	29.66 ± 8.16	26.54 ± 6.71	28.85 ± 5.24
Enhanced	26.16 ± 4.44	19.24 ± 4.54	14.41 ± 3.24
Regularized	**23.28**± 6.67	**19.01** ± 4.87	15.00 ± 3.82
Ext. Regularized	28.55 ± 8.36	25.38 ± 6.71	20.36 ± 5.33

This is probably because the training set had 80K samples, resulting in approximately 8K support vectors after SVM training, while the adaptation set only had 1K-3K adaptation samples; thus the ”extended regularized” algorithm would penalize more on the training set than on the adaptation set.

Additionally, it is interesting to compare the above with the maximum likelihood linear regression (MLLR) algorithm described in Chapter 3 for GMM adaptation, since MLLR has been the most successful in adapting Gaussian mixtures in speech recognition systems. To this end, we constructed a simple GMM classifier, where we empirically selected the maximum number of mixture components to be 32 in all 8 class-conditional Gaussian mixtures. The parameters of these Gaussians are estimated using the maximum likelihood criterion. The baseline (speaker-independent) classification error rate was 39.62%. We then implemented a native MLLR for Gaussian mean adaptation, where we let all Gaussian components in the same mixture share the same affine transform. The error rates after applying MLLR were $28.59 \pm 3.87\%$, $24.28 \pm 4.09\%$ and $20.05 \pm 3.76\%$ for 1K, 2K and 3K samples per speaker respectively. As we can see, although the GMM baseline performance was nearly as good as that of the SVM we just described, the adaptation performance of MLLR was not as effective as the regularized adaptation algorithm for SVMs.

5.6.2 Objection recognition

The recognition of generic object classes with invariance to poses and lighting conditions is one of the major challenges in computer vision [96]. In this dissertation, we apply our proposed algorithms to lighting condition adaptation, where a small number of images under a specific lighting condition are used as the adaptation data, and where the adapted classifier is used to recognize object classes under the same lighting condition. First, we describe the corpus used in our experiments.

The NORB Corpus

Here we will frequently use the terms *class*, *object* and *image*. For example, consider ”animals” as a *class*, then ”elephant”, ”tiger” and ”bear” are three different *objects* belonging to this class. For each object, there can be a good number of *images* taken under different lighting conditions and from different angles. Our corpus is a subset of the *normalized* NORB dataset [166], where the images were segmented, normalized and then composed in the center of 96x96 pixel background images. Our dataset consists of images from 5 classes, *i.e.*, airplanes, cars, trucks, human figures and animal figures, each with 10 objects (5 for training and 5 for testing). The images of each object were captured under 6 lighting conditions and with 18 poses [2] (3 different elevations and 6 azimuths). Figure 5.3 shows images of 5 objects (one per class) under different lighting conditions, but keep in mind that each example has images shot with other poses.

The training and test set each have 2,700 images, *i.e.*, 450 images per lighting condition. We performed two sets of experiments with different amounts of adaptation data. In experiment NORB-A, we adapted using images from only *one object per class*, and evaluated on the remaining images. In other words, 90 images were used as the adaptation data for each lighting condition, and the remaining 360 images were used for evaluation. We performed five-fold cross-validation, each time choosing a different object for adaptation. In experiment NORB-B, we adapted using images from *two objects per class*. In other words, 180 images were used for adaptation and 270 images for evaluation, and we performed ten-fold cross-validation. In either case, the final error rate was averaged over the number of folds. We did not further increase the size of the adaptation sets due to limited data.

[2]The images in the original NORB dataset each have 162 different poses.

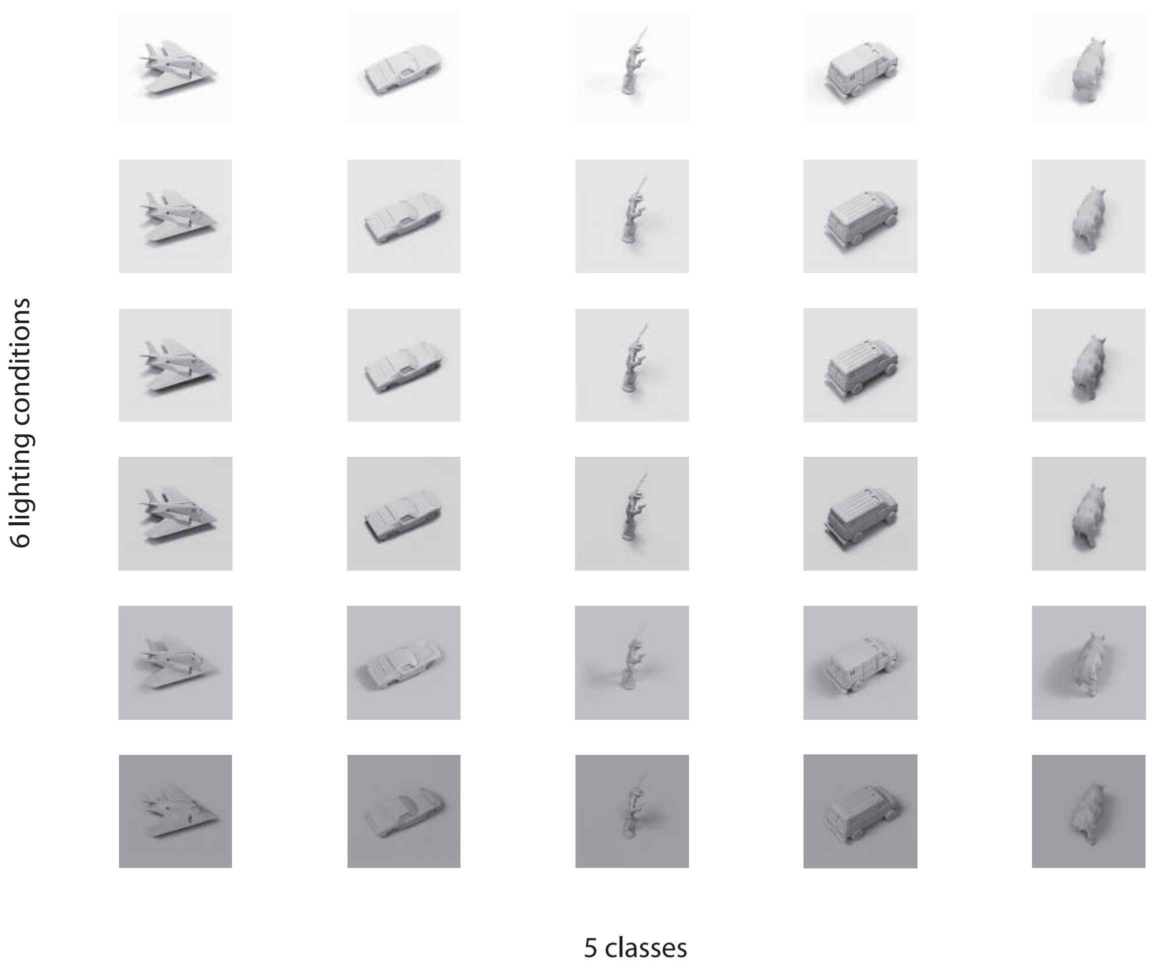


Figure 5.3: Five objects (each from a different class) under six different lighting conditions

Regarding input features, since finding pose and lighting-invariant features for object recognition is beyond the scope of our work, we simply down-sampled the images to 32x32 pixels and used raw pixel values as input features. This resolution was reported by [166] as the best setting for SVM classification. Each feature element takes integer values in the range of [0, 255], and we treated them as real numbers.

SVM Experiments

We first trained an unadapted SVM, on 2,700 training images, using Gaussian kernel with std=500. This kernel parameter was empirically chosen using cross-validation on the training set, and was fixed during adaptation. Note that using scaled inputs would change this value. The regularization coefficient C was tuned through cross-validation as well. However, we found that this dataset was well separable and that any $C > 0$ yielded the same performance; thus we simply used $C = 100$ in both training and adaptation.

We compared different SVM adaptation algorithms as were described in the vowel classification experiments, namely (1) unadapted, (2) retrained, (3) boosted, (4) enhanced, (5) regularized, and (6) extended regularized. On this dataset where the adaptation sample size was extremely small, the last strategy worked fairly well since the old support vectors therein served as extra ”adaptation data”. As shown in Table 5.7 and Table 5.8, “extended regularized” worked consistently the best for both experiment NORB-A and NORB-B, and “boosted”, which also utilizes the old support vectors, gave the second best results. In experiment NORB-B, where the amount of the adaptation data doubled, ”extended regularized” still outperformed others while “regularized” became the second best in a few cases. In addition, increasing the amount of adaptation data increased accuracy consistently.

MLP experiments

For completeness, we also tried MLP classifiers on this image dataset, although the performance was much worse than that using SVMs. We first trained an unadapted, two-layer MLP with 1024 input nodes and 5 output nodes. We varied the number of hidden units from 10 to 100 with a step size of 10, and from 100 to 500 with a step size of 50. Each experiment was accompanied by a search for the best regularization coefficients λ^{h2o} and λ^{i2h}. We found that $N = 30$ was the minimum number of hidden units that yielded the best performance, and the corresponding regularization coefficients were $\lambda^{h2o} = 5 \times 10^{-3}$ and $\lambda^{i2h} = 10^{-2}$. This unadapt MLP had an average error rate of 21.4% on the test set.

We then conducted experiments NORB-A and NORB-B using different MLP adaptation algorithms, including (1) unadapted, (2) retrained (from scratch) with weight decay, (3) RSI (from the unadapted model), and (4) regularized. Here we ignored LIN since intuitively it was likely to cause

Table 5.7: Experiment NORB-A using SVMs with a Gaussian kernel (std=500) (90 images per lighting condition). The tradeoff coefficient is $C = 100$ in all cases. The highlighted entries in **avg** include the best error rate and those **not** significantly different from the best at the $p < 0.001$ level.

Lighting	0	1	2	3	4	5	avg
Unadapted	11.1	11.3	16.5	10.9	15.8	9.3	12.5
Retrained	30.9	24.5	36.4	29.3	29.1	30.6	30.1
Boosted	10.3	10.6	14.9	10.4	13.3	9.3	**11.5**
Enhanced	21.8	16.9	31.5	20.6	23.1	17.6	21.9
Regularized	13.2	13.3	17.1	13.7	16.6	11.9	14.8
Ext. Regularized	9.8	9.8	14.1	10.1	12.5	9.4	**11.0**

Table 5.8: Experiment NORB-B using SVMs with a Gaussian kernel (std=500) (180 images per lighting condition). The tradeoff coefficient is $C = 100$ in all cases. The highlighted entries in **avg** include the best error rate and those **not** significantly different from the best at the $p < 0.001$ level.

Lighting	0	1	2	3	4	5	avg
Unadapted	11.1	11.3	16.5	10.9	15.8	9.3	12.5
Retrained	18.2	14.5	22.7	19.3	20.6	18.4	18.9
Boosted	9.9	10.2	13.6	10.1	11.5	9.1	**10.7**
Enhanced	15.4	11.8	19.2	15.6	15.6	12.0	14.9
Regularized	11.4	11.2	14.7	12.3	14.6	16.4	13.4
Ext. Regularized	8.8	9.2	11.8	9.9	14.0	8.7	**10.4**

overfitting by estimating a 1024×1024 transformation matrix. In choosing the tradeoff coefficients λ^{h2o} and λ^{i2h} for the "regularized" experiments, we found that many values gave statistically identical performance, and we chose the highest such values in order to maximize the degree of regularization. As shown in Table 5.9 and 5.10, RSI and regularized worked identically (statistically) the best in both experiment sets.

Table 5.9: Experiment NORB-A using 5-class MLPs (90 images per lighting condition). The highlighted entries in **avg** include the best error rate and those **not** significantly different from the best at the $p < 0.001$ level.

Lighting	0	1	2	3	4	5	avg
Unadapted	20.1	23.4	34.2	21.2	15.8	13.7	21.4
Retrained (λ^{h2o}=0, λ^{i2h}=0)	35.0	34.2	77.1	37.4	33.4	34.5	41.9
RSI (λ^{h2o}=0, λ^{i2h}=0)	17.4	20.8	28.5	18.3	16.8	15.8	**19.6**
Regularized (λ^{h2o}=∞, λ^{i2h}=10^{-2})	17.6	20.7	26.1	18.3	16.5	15.9	**19.2**

Table 5.10: Experiment NORB-B using 5-class MLPs (180 images per lighting condition). The highlighted entries in **avg** include the best error rate and those **not** significantly different from the best at the $p < 0.001$ level.

Lighting	0	1	2	3	4	5	avg
Unadapted	20.1	23.4	34.2	21.2	15.8	13.7	21.4
Retrained ($\lambda^{h2o} = 0$, $\lambda^{i2h} = 0$)	28.3	27.2	61.0	27.2	25.2	26.0	32.5
RSI (λ^{h2o}=0, λ^{i2h}=0)	16.3	20.0	27.5	17.4	15.7	14.7	**18.6**
Regularized (λ^{h2o}=∞, λ^{i2h}=10^{-2})	16.4	19.3	25.6	17.0	15.1	14.9	**18.0**

Chapter 6
APPLICATION TO THE VOCAL JOYSTICK

In Chapter 4 and Chapter 5, we presented a principled approach to adaptation and discussed in detail the derived algorithms for GMMs, SVMs and MLPs. The motivation of conducting such research was originated from the development of the Vocal Joystick — a voice based human-computer interface for individuals with motor impairments.

In Chapter 1, we discussed the importance of developing voice-based computer interfaces that are capable of controlling continuous tasks. The Vocal Joystick is such an interface developed at the University of Washington [36, 37]. Unlike standard ASR, our system goes beyond the capabilities of sequences of discrete speech sounds, and exploits continuous vocal characteristics such as intensity, pitch and vowel quality which are then mapped to continuous control parameters This chapter describes the Vocal Joystick engine as well as the role of adaptation in the VJ system. Specifically, Section 6.1 overviews four major acoustic parameters in the VJ system; Section 6.2 describes engine organization, with focus on the signal processing and pattern recognition modules. The last section discusses why adaptation is important and how our proposed algorithms are applied.

6.1 Overview of Acoustic Parameters

The key characteristic of the Vocal Joystick is that unlike conventional ASR systems, which recognize sequences of words, it processes continuous acoustic parameters every audio frame and transforms them into various control parameters. There are four major acoustic parameters currently used in the VJ system, *intensity*, *pitch*, *vowel quality* and *discrete sound identity*.

Intensity, computed as the average energy of an acoustic signal, is a measure of ”loudness”. The average energy, along with zero-crossings, plays an important role in voice activity detection (VAD) and determines the start-and-end of a mouse movement. Intensity is also used to control the movement speed. For example, a loud voice corresponds to a large movement while a quiet voice corresponds to a nudge.

The second parameter is pitch. Pitch is the auditory percept of tone, and is often calculated as the inverse of the smallest period of an acoustic signal. Currently pitch is not used in the VJ-mouse system, but is potentially useful in systems that require more degrees of freedom in control, *e.g.* an robotic arm with three joints.

The third parameter is vowel quality. Vowels are voicing phonemes generated by vibration of the vocal folds [167]. They are the most intense and the most audible sounds in speech. They usually function as syllable nuclei, carrying information of energy and pitch. Consonants, on the other hand, are speech phonemes produced by constriction of the passage through the throat and the mouth, which are in general less audible than vowels. The use of vowels, therefore, may lead to a classification system with high discriminability and noise-robustness. In this work, vowel quality is used to control the direction of a mouse movement. we choose to use four vowels /æ/, /ɑ/, /u/ and /i/ (in the IPA alphabet [168]) to control four cardinal directions in a 2-D space (up, left, down and right), as illustrated in Figure 6.1. Furthermore, four additional vowels, /a/, /o/, /ɨ/ and /e/, are designated for diagonal directions. The ”vowel triangle”, depicted in Figure 6.2, shows the articulatory properties of these vowels in terms of tongue positions in the mouth. The vowel triangle roughly reflects the *relative* locations of these vowels in acoustic space. We chose to use /æ/, /ɑ/, /u/ and /i/ for cardinal directions because they are at the four corners of the vowel triangle and are presumably the most far apart in acoustic space.

Finally, certain combined vowels and consonants, which we call discrete sounds, can be used to control discrete actions, e.g., mouse clicks and toggles. In this regard, discrete sounds are akin to voice commands except that they are not confined to be verbal. In fact, they can be chosen in a way that maximally reduces confusability. This can be achieved by incorporating phonetic knowledge, *e.g.*, the articulation similarity of different phonemes, or model knowledge since the confusion patterns may depend on the classifier.

6.2 The VJ Engine

Utilizing the acoustic parameters introduced above, we have developed a VJ-mouse control scheme which works as follows: mouse movement is triggered when vowel activity is detected, and continues until the vowel activity stops. At each time frame, the movement direction is determined by

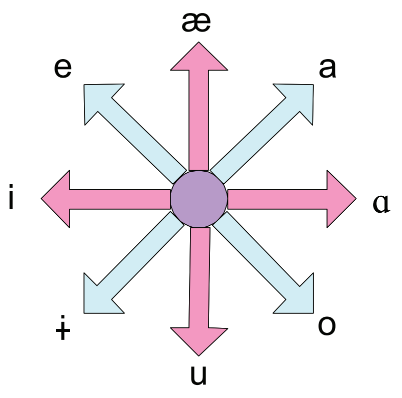


Figure 6.1: Vowel-direction mapping

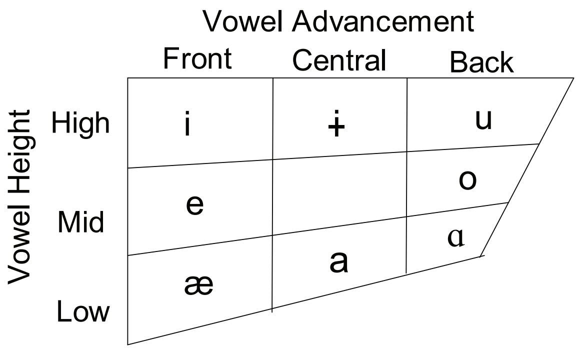


Figure 6.2: Vowel triangle

vowel quality, while the step size is determined by intensity. Finally, a small set of discrete sounds are used to execute mouse clicks. Now we present the VJ engine in more detail. A general picture of the VJ engine is shown in Figure 6.3, which consists of three main components: signal processing, pattern recognition and motion control.

6.2.1 Signal processing

The goal of the signal processing module is to perform voice activity detection (VAD), and to extract low-level features such as average energy, number of zero-crossings, normalized autocovariance coefficients (NACCs) and mel frequency cepstral coefficients (MFCCs) [169, 170], all of which are

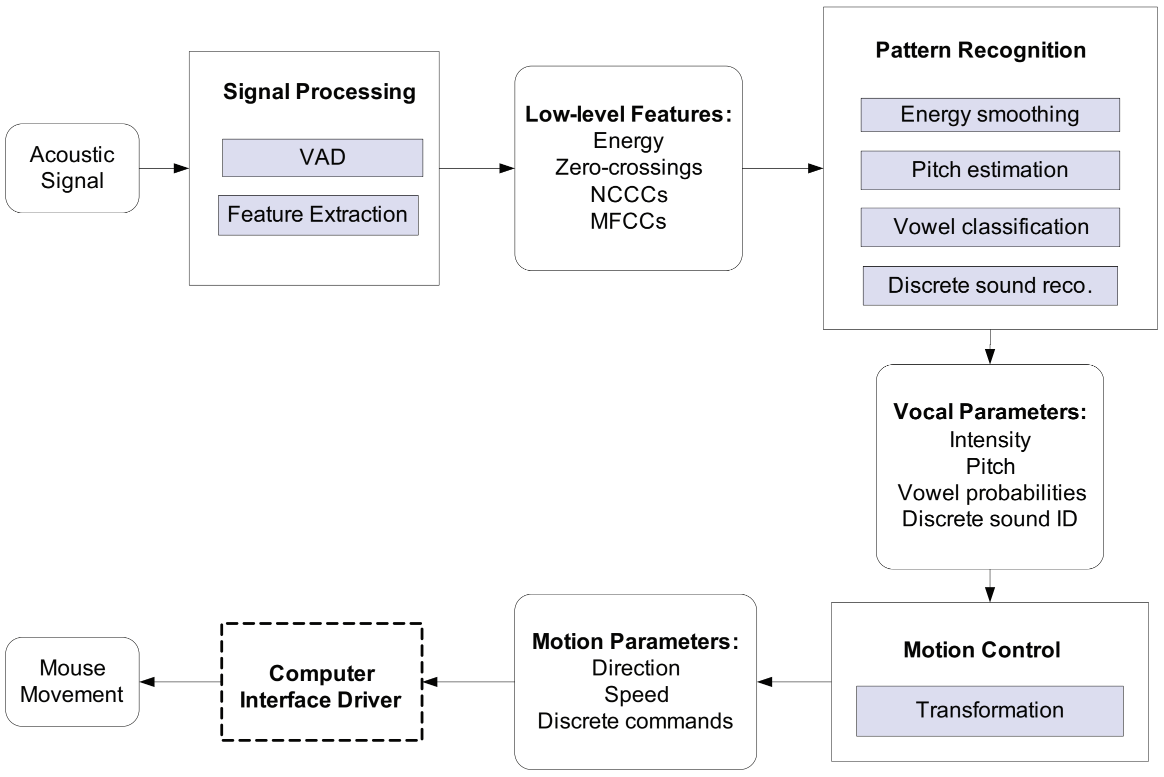


Figure 6.3: The VJ engine architecture. This dissertation mainly contributes to the signal processing and pattern recognition module.

inputs to the pattern recognition module. To this end, we sample an incoming acoustic signal at a rate of 16kHz, and generate a frame every 10ms using a 40ms rectangular window. We choose such a window length only to guarantee that it is at least twice of the longest possible pitch period, as suggested by [169] — since the pitch of a normal human voice ranges between 50Hz and 500Hz, we use a window length of 20ms $\times$ 2 $=$ 40ms. Such a choice makes it relatively easy to obtain reliable pitch estimates, but it is entirely possible to use a shorter window. Note that this window is used for computing all low-level features except MFCCs which use a 25ms Hamming window instead.

Let $s_t(l)$ denote sample l of frame t, where $l = 1..L$ and $L =$ 16kHz $\times$ 40ms $=$ 640 is the rectangular window length. For each frame, we first compute the average energy as

$$e_t = \frac{1}{L}\sum_{l=1}^{L} |s_t(l)|^2 \tag{6.1}$$

and the number of zero-crossings as

$$z_t = \sum_{l=1}^{L-1} \mathrm{I}(s_t(l)s_t(l+1) < 0) \tag{6.2}$$

Both e_t and z_t are strong indicators of voice activity — the larger the values of e_t and z_t, the more likely voice activity occurs. We then use a set of empirical thresholds to categorize a frame into ”active”, ”pre-active”, ”temporarily inactive” and ”truly inactive” as follows.

- Frame t is **active** if $e_t \geq a_h e_{\text{noise}} + e_o$ and $z_t > z_o$, where e_{noise} is an estimate of the background noise level, a_h is a constant multiplier, and e_o and z_o are constant floor values. An active frame is believed to contain voice activity, for which we extract low-level features such as NACCs and MFCCs, and for which we conduct pattern recognition functions based on the extracted features of the current frame and sometimes of previous frames as well.

- Frame t is **pre-active** if $a_l e_{\text{noise}} + e_o < e_t < a_h e_{\text{noise}} + e_o$ and $z_t > z_o$, where a_l is a constant multiplier and $a_l < a_h$. All low-level features are extracted in such a state, but no pattern recognition functions are actuated. The main purpose of creating the pre-active state is to indicate that the frame following a pre-active frame might be active and hence low-level features of the pre-active frame need to be computed and cached for potential use in the near future. Note that ”pre-active” is only necessary in a real-time system.

- Frame t is **temporarily inactive** if $e_t \leq a_1 e_{\text{noise}} + e_o$ or $z_t \leq z_o$. However, we still generate low-level features (again without executing pattern recognition functions) until the number of consecutive inactive frames exceeds a threshold. Only at this point do we believe that the voice activity has fully stopped, and do we call it a **truly inactive** state; otherwise the observed ”temporarily inactive” frames might merely correspond to a pause in an utterance, which we should not consider as the end of a voice activity. Once the number of consecutive temporarily inactive frames exceeds the threshold, we stop generating NACCs and MFCCs, and we send the the cached low-level features of the proceeding voice activity segment to a discrete sound recognizer.

As stated above, we extract NACCs and MFCCs when a frame is active, pre-active, or temporarily inactive. NACCs are input features to the pitch tracker in the pattern recognition module. For a better understanding of these features, we first introduce *autocorrelation coefficients* [169, 170], which show how well a signal correlates with itself at a range of different delays, and we expect a periodic signal has the highest correlation with itself at the pitch period. Mathematically, the autocorrelation coefficient for lag $i > 0$ is defined as

$$r_{t,i} = \frac{1}{L-i} \sum_{l=1}^{L-i} s_t(l) s_t(l+i)$$

The autocorrelation method is an unbiased estimator, but the variance gets larger as i gets closer to L [169–171]. Therefore, we often need to use an excessively large window for short lags to maintain significance for large lags. To compensate for this problem, *autocovariance coefficients* [1] [42, 169] use an extended window $L_{\text{ext}} > L$ such that the summation interval does not shrink for large lags. Here we use $L_{\text{ext}} = L + i_{\text{max}} = 640 + 320 = 960$, where $i_{\text{max}} = 320$ is the maximum lag (corresponding to 50Hz). NACCs, which are utilized in [172] and the current VJ engine, are simply energy-normalized autocovariance coefficients. Mathematically,

$$\phi_{t,j} = \frac{\sum_{l=1}^{L} s_t(l) s_t(l+j)}{\sqrt{e'_{t,0} e'_{t,j}}} \tag{6.3}$$

where $e'_{t,j} = \sum_{l=1}^{L} |s_t(l+j)|^2$, and it is easy to see that in Equation (6.1) $e_t = e'_{t,0}$.

Finally, MFCCs and their dynamic features are extracted using a standard method described in [42]. Note again that we replace the 40ms rectangular window by a 25ms Hamming window. These features are inputs to the vowel classifier and the discrete sound recognizer in the pattern recognition module.

[1]Auto-covariance is also called modified autocorrelation or sometimes cross-correlation, but the latter is a misnomer as cross-correlation usually refers to the correlation between two signals

6.2.2 Pattern recognition

The pattern recognition module serves as an interface which takes the above low-level features as input and estimates acoustic parameters we introduced in Section 6.1, including intensity, pitch, vowel quality posterior probabilities and discrete sound identity, as shown in Figure 6.3. In the following text we will discuss in detail how these acoustic parameters are estimated, but we will leave to Chapter 7 the discussion of a novel pitch estimation algorithm, and an online adaptive filter algorithm applied to the vowel classifier outputs.

First of all, intensity is simply a smoothed version of the average energy e_t, which is obtained by applying a low-pass FIR filter on e_t.

Next, the pitch trajectory is estimated using NACCs. Specifically, the current VJ engine selects local pitch candidates for frame t based on $\phi_{t,i}$ in Equation (6.3). — if $\phi_{t,\cdot}$ achieves its highest value at i_0, it is likely that i_0 corresponds to the true pitch period. Local estimates, however, are not always reliable. This is because multiples of the pitch period can also yield high $\phi_{t,\cdot}$ values, sometimes leading to pitch halving errors; on the other hand, pitch harmonics can cause pitch doubling errors. A more reliable method is to apply dynamic programming, which takes into account context information to obtain globally optimal estimates [172]. Many pitch trackers based on dynamic programming require meticulous design of local cost and transition cost functions. The forms of these functions are often empirically determined and their parameters are tuned accordingly. Parameter tuning usually requires great effort without a guarantee of optimal performance. Chapter 7 will present a graphical model framework to automatically learn pitch tracking parameters in the maximum likelihood sense.

Given the pitch estimate, we can immediately decide whether a frame is voiced or unvoiced. For each voiced frame, we perform vowel classification using a two-layer MLP with 50 hidden nodes as described in the previous chapter. The MLP takes 7 consecutive frames of MFCCs and their dynamic features as input, denoted as $\mathbf{x}_t$, and outputs a vowel posterior probability vector $\mathbf{z}_t$ where $z_{t,k} = p(k|\mathbf{x}_t)$ is the posterior probability of vowel class k. In one configuration, we use a 4-class MLP, corresponding to /æ/, /ɑ/, /u/ and /i/, whereas a second configuration uses a 8-class MLP which include the other four vowel classes /a/, /o/, /ɨ/ and /e/. These MLP classifiers were trained on the VJ-vowel corpus described in Chapter 5. In Chapter 7, we will discuss an online adaptive filter algorithm applied to $\mathbf{z}_t$. This algorithm utilizes context information and enables the VJ system to produce movements in arbitrary directions.

Lastly, we launch a discrete sound recognizer when we have detected the end of voice activity (by inspecting the number of consecutive temporarily inactive frames, as mentioned in the signal processing module). Discrete sound recognition is essentially a problem of isolated-word recognition except that in our case we can compose any "word" using a set of phonemes. The current VJ engine adopts a vocabulary that only contains unvoiced consonants, each defined as a different discrete sound. The purpose that we do not include any voiced phonemes in the discrete sounds is to maximally reduce confusion with the vowels used in continuous control. For example, using "open" as a discrete sound would cause a continuous movement since it starts with the vowel /o/; even using discrete sounds such as /kə/ can be problematic occasionally, as in user studies we have observed situations where the starting /k/ was indiscernible or was simply omitted and the following /ə/ was recognized as a vowel resulting in a mouse movement.

Given a vocabulary of discrete sounds, our goal is to find the one that best matches an input signal. Letting $\mathbf{x}_{1:T}$ denote the input features and k the discrete sound identity, we want to find

$$\hat{k} = \underset{k}{\operatorname{argmax}}\ p(k|\mathbf{x}_{1:T}) = \underset{k}{\operatorname{argmax}}\ p(\mathbf{x}_{1:T}|k)p(k). \tag{6.4}$$

Here we use Gaussian-mixture HMMs to represent $p(\mathbf{x}_{1:T}|k)$, and we train these HMMs on TIMIT [165] — a phonetically-rich, continuous speech corpus with time-alignment information. The utterances in TIMIT were sampled at 16kHz which is compatible with the setting of the VJ system. We defined 42 phonemes, with 3 states per phoneme and 12 Gaussian components per state, leading to 126 Gaussian mixtures. MFCCs and their deltas were extracted using the VJ frontend and were input to HTK [173] for HMM estimation.

The above formulation is based on the assumption that $\mathbf{x}_{1:T}$ is the acoustic signal of an in-vocabulary discrete sound. However, it is rather common that some inputs do not fall within the vocabulary that the recognizer is designed to handle. An out-of-vocabulary utterance may consist of a vowel articulation (intended for a continuous movement), extraneous speech or background noise. A rejection mechanism, therefore, is indispensable to the reliable functioning of the discrete sound recognizer. Numerous strategies for rejecting garbage utterances in isolated word recognition have been proposed in the literature, including the use of filler models [174,175] and the use of confidence measures [176, 177]. However, we found the filler-model approach less effective for the VJ engine. The main reason is that many false positives in our system are caused by breathing, coughing, spike noise or other random unvoiced-consonant-like sounds (since our vocabulary consists of pure unvoiced consonants). In most cases, these random sounds match one of the vocabulary items better than the filler model (which in our case was trained on TIMIT — a speech corpus containing a wide variety of vowels and consonants). On the other hand, the use of confidence measures alone is not stable. Here we describe several complementary rejection mechanisms designed specifically for the VJ engine.

- We make a first pass of rejection by discarding an utterance whose **duration** is out of a pre-defined range, where the range is vocabulary-dependent and is determined empirically.

- The second rejection mechanism is strongly related to the choice of the vocabulary. We currently use one unvoiced consonant per discrete sound, *e.g.*, /ch/ and /t/. Based on this assumption, we impose thresholds on the **average number of zero-crossings** and on the **portion of unvoiced frames** in an utterance. The utterance is rejected if these numbers are below their respective pre-defined thresholds.

- The above two rejection mechanisms have significantly reduced false positives caused by vowel articulation, but are still vulnerable to unvoiced-consonant-like sounds such as breathing. Thus we create a confidence measure using the **posterior probability** of the best model candidate. Specifically, we accept an utterance only if $p(\hat{k}|\mathbf{x}_{1:m})$ is above a threshold, where $\hat{k}$ is defined by Equation (6.4).

6.2.3 Motion control

Finally, the motion control module determines how to transform these acoustic parameters into motion parameters such as direction and speed. For the VJ-mouse application, we produce a 2-D motion vector $\mathbf{v}_t = (\Delta x, \Delta y)^T$ (relative movement) for each vowel frame, where $\|\mathbf{v}_t\| = \sqrt{\Delta x^2 + \Delta y^2}$ is determined by intensity and $\angle \mathbf{v}_t$ is determined by vowel quality. How to map intensity to the norm of the motion vector has been investigated in [53]; here we focus on mapping vowel quality to the angle of the motion vector, which is necessary to understanding our proposed online adaptive filter algorithm in Chapter 7. In other words, we assume $\Delta x^2 + \Delta y^2 = 1$.

A *soft classification* strategy is adopted to map vowel quality to movement direction. We categorize the vowel space and assign directions to vowel classes as shown in Figure 6.1. Given a vowel posterior probability vector $\mathbf{z}_t$ output by the MLP classifier, we map $\mathbf{z}_t$ to a unit motion vector using a linear transformation defined as follows

$$\begin{aligned} \Delta x &= \langle \mathbf{z}_t, \mathbf{w}_x \rangle \\ \Delta y &= \langle \mathbf{z}_t, \mathbf{w}_y \rangle \end{aligned} \tag{6.5}$$

where $\mathbf{w}_x$ and $\mathbf{w}_y$ are two weight vectors whose elements satisfy $\mathbf{w}_{x,i} = \cos\frac{2\pi i}{n}$ and $w_{y,i} = \sin\frac{2\pi i}{n}$; here n is the number of vowel classes, and the vowel categories $i = 0, 1, \ldots, n-1$ are indexed counterclockwise with respect to Figure 6.1, starting from /ɑ/.

6.3 Application of Regularized Adaptation Algorithms

In the VJ system, vowel quality and discrete sound identity are used for continuous and discrete control respectively. Therein, inaccuracies can arise due to speaker variability owing to different vocal tract lengths and speaking styles. A vowel class articulated by one user might partially overlap in acoustic space with a different vowel class from another user. This imposes limitations on a purely user-independent vowel classifier. The problem is even more pronounced for the discrete sound recognizer. This is because our current user-independent discrete sound recognizer was trained on a continuous speech corpus in which consonants are often articulated in the context of vowels, while our target application only anticipates unvoiced consonants spoken in an isolated manner. This mismatch can severely degrade the recognition performance. To mitigate these problems, we have designed an adaptation procedure where each user is asked to pronounce a small number of adaptation samples. The current system asks for 200 samples per vowel class (corresponding to a 2-second utterance for each vowel) and one utterance per discrete sound, but these numbers are reconfigurable. Figure 6.4 shows the user interface for adaptation.

Once the adaptation data is collected, we launch the regularized adaptation algorithms for the MLP-based vowel classifier and for the HMM-based discrete sound recognizer. MLP adaptation uses the update formula in Equation (5.32). For HMM adaptation, due to limited adaptation data,

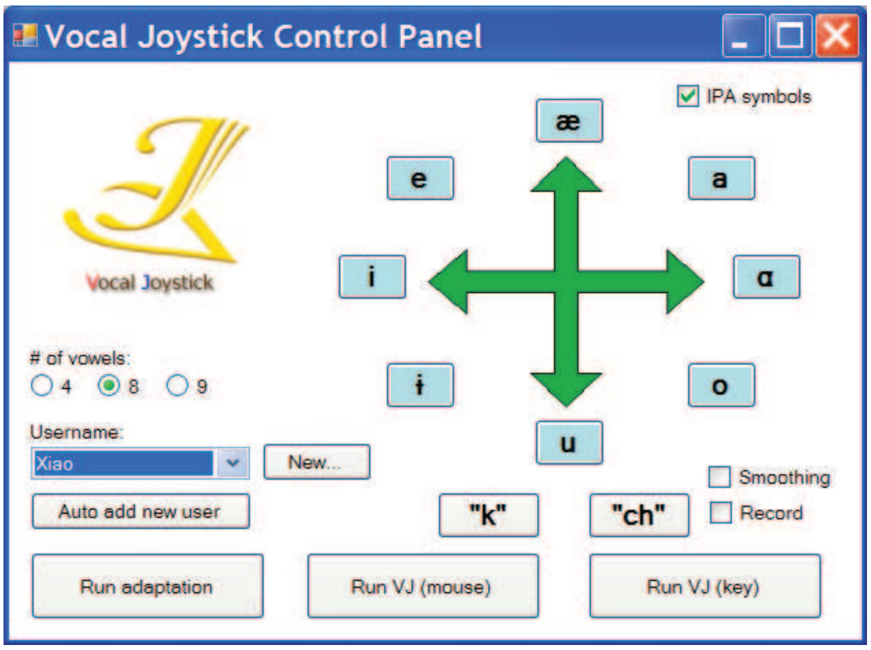


Figure 6.4: An interface for VJ adaptation

we only adapt Gaussian means according to the update formula in (5.7). Along with the collection of vowel data, we also compute the average intensity for each vowel, and use these intensity values as the intensity-to-speed normalization factors in the motion control module [53]. Similarly, along with the collection of discrete sound data, we compute the utterance length and the number of zero-crossings for each vocabulary item, and utilize these values for adapting the discrete sound rejection system.

With the aid of adaptation, the VJ system has become more accurate, as supported by the evidence in the vowel classification experiments in Chapter 5, and more reliable to use, as shown in a number of user studies [36, 37, 54]. Our website http://ssli.ee.washington.edu/vj/video_demos.htm has a dozen of video clips that demonstrate using the VJ-mouse to accomplish various real-world tasks, such as browsing a website or playing a computer game.

Chapter 7
OTHER MACHINE LEARNING TECHNIQUES IN THE VOCAL JOYSTICK

The previous chapter presented the VJ engine architecture. There are two components in the pattern recognition module yet to be explained in this chapter. The first component is a pitch tracker that estimates the pitch trajectory of an acoustic signal. Section 7.1 will present a Bayesian network that automatically optimizes pitch tracker parameters in the maximum likelihood sense. This framework not only expedites the training of the pitch tracker, but also yields remarkably good performance for both pitch estimation and voicing decision. The second component, as will be presented in Section 7.2, involves an online adaptive filter applied to the vowel classifier outputs. This algorithm utilizes context information and applies real-time inference in a continuous space. The VJ system using this algorithm is endowed with the ability to produce movements in arbitrary directions and the ability to draw smooth curves. This is in contrast to previous VJ settings whereby vowel quality was used to determine mouse movement in only a finite discrete set of directions.

7.1 Pitch Tracking

Pitch tracking has drawn increased attention in speech coding, synthesis and recognition. In the VJ system, pitch can be utilized to provide an additional degree of freedom. Developing a robust pitch tracker, therefore, is important to the development of potential VJ-based applications such as VJ-robotic-arms.

Many state-of-the-art pitch trackers resemble the methodology proposed by [172], which consists of three steps: pre-processing, pitch candidate generation and post-processing by dynamic programming (DP). The first step involves signal conditioning techniques. The second step selects pitch candidates and computes their "scores" by applying certain pitch detection algorithms (PDA) to the local frame acoustics. In the post-processing step, the cost $C_{t,j}$ of proposing pitch candidate j at frame t is computed as follows,

$$C_{t,j} = F_t^{local}(j) + \min_i \{C_{t-1,i} + F_{t-1,t}^{tran}(i,j)\} \tag{7.1}$$

where the local cost function F^{local} takes into account the scores obtained from the second step, and the transition cost function F^{tran} models the penalty of transitioning from candidate i of the previous frame to candidate j of the current frame.

The forms of these cost functions are usually empirically determined and their parameters are often tuned by algorithms such as gradient descent [172]. This process, however, remains a difficult and time-consuming task. First, F^{local} has to be optimized each time a different PDA is applied. For example, PDAs can be designed in several domains including time, spectral, cepstral and their combinations [178]. While classic PDAs like the normalized autocovariance coefficients (NACCs) [179] are popularly used, new algorithms such as ACOLS [180], JTFA [181] and YIN [182], are increasingly coming into play. In order to evaluate, compare and eventually implement these techniques, a large amount of time has to be spent deciding the form of F^{local} and tuning its parameters. Second, ideally F^{tran} should be adapted when the pitch tracker is exposed to another language, or applied to another application. This is because different languages and applications may follow very different pitch transition patterns. Therefore, the local and transition cost functions optimized for certain PDAs and applications may not be the most appropriate for others.

This section presents a strategy for optimizing the parameters of these cost functions using a more principled approach. Extending the idea of [183], we present a graphical model framework to learn a pitch tracker from data. Therein, a PDA or a pitch transition pattern can be easily incorporated into the system with parameters automatically estimated using statistical methods. Furthermore, since the parameters are optimized in the maximum likelihood sense, both pitch estimation and voicing decision give better performance. The Graphical Model Toolkit (GMTK) [184] was utilized to implement our framework.

7.1.1 Graph structure and local probability models

Graphical models are a flexible, concise, and expressive probabilistic modeling framework with which one may rapidly specify a vast collection of statistical models. Our graphical model framework for pitch tracking is depicted in Figure 7.1, where the shaded nodes represent variables observed at **decode** time and the unshaded nodes are hidden. These variables are defined as follows.

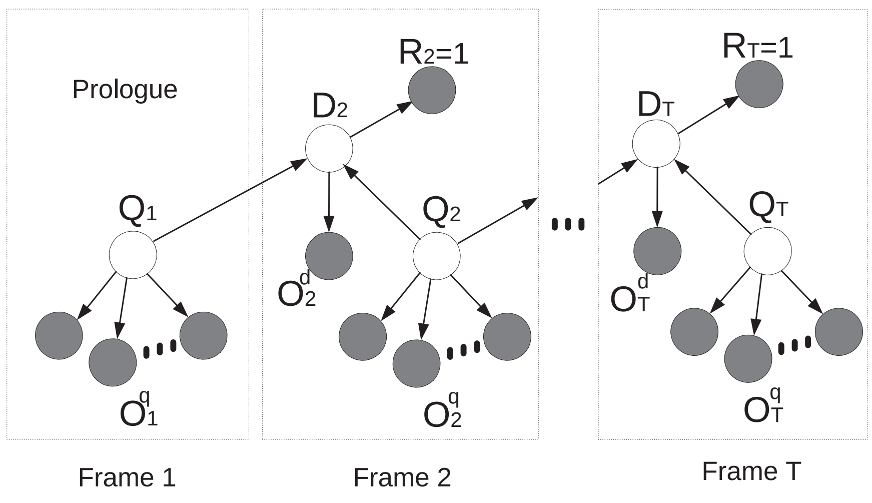


Figure 7.1: Decoding graph

- Variable Q_t is discrete with cardinality N, which corresponds to $N-1$ voiced states (indexed as $1..N-1$) plus one unvoiced state (indexed as N) at frame t. The $N-1$ voiced states are determined by $N-1$ possible pitch periods as explained in Chapter 6. In the graph, Q_t has no parents, but has a prior distribution $\pi_i^q \triangleq P(Q_t = i)$.

- Variable D_t is discrete with cardinality M, corresponding to M transition patterns coarsely quantized from N^2 possible (Q_{t-1}, Q_t) pairs. The dependency between Q_{t-1}, Q_t and D_t is represented by a deterministic table. Specifically, the set $\{1..N\} \times \{1..N\}$ is partitioned into M non-overlapping subsets $\mathcal{S}_m$, $m = 1..M$, and we define $P(D_t = m|Q_{t-1} = i, Q_t = j) = 1$, Iff $(i, j) \in \mathcal{S}_m$. In this work, we use a simple partition scheme:

$$\begin{aligned}
\mathcal{S}_1 &= \{(i,j) : i = N, j \neq N\}; \\
\mathcal{S}_2 &= \{(i,j) : i \neq N, j = N\}; \qquad (7.2) \\
\mathcal{S}_m &= \{(i,j) : L_m \leq j - i < U_m\}; \quad m = 3..M,
\end{aligned}$$

where L_m and U_m are (respectively) lower and upper bounds evenly spaced at integers between $-N+2$ and $N-2$. In other words, $\mathcal{S}_1$ corresponds to unvoiced-to-voiced transitions; $\mathcal{S}_2$ corresponds to voiced-to-unvoiced transitions; $\mathcal{S}_m$, $m = 3..M$ correspond to pitch transitions (or voiced-to-voiced transitions) clustered into $M-2$ patterns based on pitch period difference; and the unvoiced-to-unvoiced transition belong to the same subset as pitch transitions where $i = j$.

- There is a dummy binary variable R_t with a parent D_t. The conditional probability π_m^d for this dependency is defined as

$$\pi_m^d \triangleq P(R_t = 1 | D_t = m) = \frac{\sum\limits_{(i,j)\in\mathcal{S}_m} \#(i,j)}{\sum\limits_{m=1}^{M} \sum\limits_{(i,j)\in\mathcal{S}_m} \#(i,j)}, \tag{7.3}$$

  where $\#(i,j)$ is the number of occurrences of event $\{Q_{t-1} = i, Q_t = j\}$ in the training data. The purpose of this dummy node is to provide soft evidence [185, 186] for D_t, and this evidence is encoded using the histogram of the M pitch transition patterns. Note that for the purposes of inference and decoding, the results should be identical with a π_m^d multiplied by any positive scalar. We keep this expression of soft evidence, as it is amenable to standard smoothing methods (see the Subseciton 7.1.3).

- Finally, variables O_t^q and O_t^d are continuous, and are children of Q_t and D_t respectively. They are both computed directly from an acoustic signal, which will be discussed in the next.

In graphical model semantics, Figure 7.1 captures dependencies between pitch values and local acoustics, and between pitch transition patterns and acoustical changes. Also, by modelling Q_{t-1} and Q_t as parents of D_t and adding soft evidence R_t, the prior probabilities of pitch and pitch transition are **simultaneously** modeled in the graph, which would otherwise be hard to accomplish.

7.1.2 Observation features

The observation features O_t^q are crucial to the success of a pitch tracking algorithm. As discussed in the previous chapter, autocorrelation coefficients or their extended forms [179,180,182] are effective features which result in time-domain PDAs. For example, in the case of NACCs, we let $\phi_t = (\phi_{t,1}, \phi_{t,2}, \ldots, \phi_{t,N})^T$ where $\phi_{t,i}$, $i = 1..N-1$, is the NACC of the i^{th} candidate pitch period, and $\phi_{t,N} \triangleq \max\limits_{i=1..N-1} \phi_{t,i}$. If frame t is voiced and the i^{th} candidate corresponds to the true pitch period, then $\phi_{t,i}$ is likely to have a high value (close to one) according to Equation (6.3). On the other hand, if frame t is unvoiced, then $\phi_{t,N}$ is likely to be small since all $\phi_{t,i}$, $i = 1..N-1$, tend to be small. We represent such relationships using Gaussian distributions as follows.

$$P(O_t^q = \phi_t | Q_t = i) = \begin{cases} \mathcal{N}(\phi_{t,i}; 1, \beta^2) & i = 1..N-1 \\ \mathcal{N}(\phi_{t,i}; \mu, \gamma^2) & i = N \end{cases} \tag{7.4}$$

where μ is the minimum value of $\phi_{t,N}$ in all training data. Note that these two means, 1 and μ, are set in advance and are fixed during the training of other parameters. Since $\phi_{t,i} < 1$, $i = 1..N-1$, a high $\phi_{t,i}$ will lead to a high observation probability for state i. Similarly since $\phi_{t,N} > \mu$, a low $\phi_{t,N}$ implies a high observation probability for state N, meaning that frame t is likely to be unvoiced.

The observation feature O_t^d is the delta energy $\Delta e_t = e_t - e_{t-1}$. The choice of this feature is based on the empirical observation (justified by our experiments) that there is a correlation between pitch transition and delta energy. For example, an utterance with decreasing pitch tends to have a decreasing energy, and an unvoiced-to-voiced transition tends to have an increasing energy. Therefore, the delta energy to some extent reflects pitch transition patterns and can improve pitch estimation. The corresponding observation distribution is modeled as

$$P(O_t^d = \Delta e_t | D_t = m) = \mathcal{N}(\Delta e_t; \rho_m, \sigma^2), \tag{7.5}$$

where ρ_m is the mean of the Gaussian of the m^{th} transition pattern, and σ^2 is a variance variable shared by all Gaussians.

7.1.3 Parameter estimation and decoding

The graph structure for **training** differs slightly from that for decoding. As shown in Figure 7.2, all variables in training are observed except D_t. In fact, D_t can be considered ”observed” as well because its value is deterministic given Q_{t-1} and Q_t. This graph structure implies several conditional independence relationships that enable the decomposition of the complete likelihood in training.

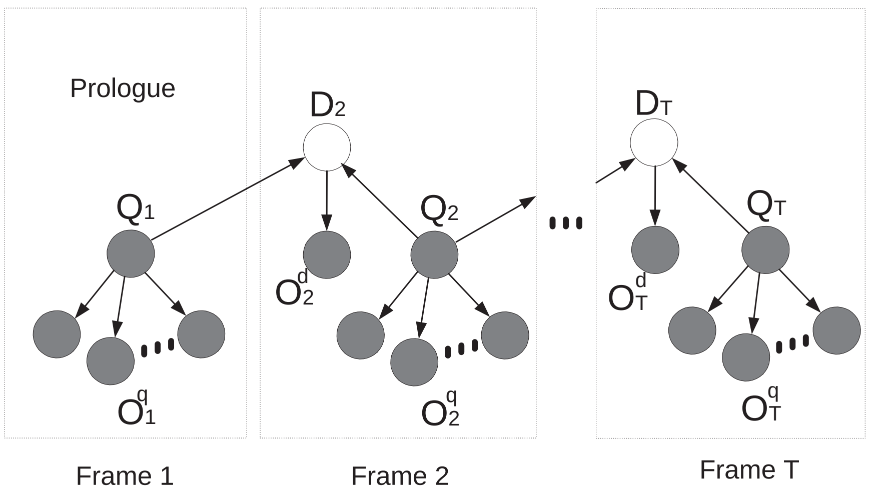


Figure 7.2: Training graph

Mathematically, we desire to learn parameters that maximize

$$\begin{aligned}
& \ln P(Q_{1:T}, O^q_{1:T}, O^d_{2:T}) \\
= \; & \ln p(Q_1, O^q_1) + \sum_{t=2}^{T} \Big[\ln p(Q_t, O^q_t) + \ln p(O^d_t | Q_{t-1}, Q_t) \Big] \\
= \; & \sum_{t=1}^{T} \ln P(Q_t, O^q_t) + \sum_{t=2}^{T} \ln \Big[\sum_{m=1}^{M} P(O^d_t, D_t = m | Q_{t-1}, Q_t) \Big] \\
= \; & \sum_{t=1}^{T} \ln P(Q_t = q_t) + \sum_{t=1}^{T} \ln P(O^q_t = \phi_t | Q_t = q_t) \\
& + \sum_{t=2}^{T} \ln [\sum_{m=1}^{M} P(O^d_t = \Delta e_t | D_t = m) P(D_t = m | Q_{t-1} = q_{t-1}, Q_t = q_t)]
\end{aligned} \tag{7.6}$$

Recall that $P(D_t = m|Q_{t-1} = q_{t-1}, Q_t = q_t)$ is an indicator function which equals one iff $(q_{t-1}, q_t) \in \mathcal{S}_m$. Therefore, only the corresponding pitch transition pattern can survive the summation over m. Plugging Equation (7.4) and Equation (7.5) into Equation (7.6) and taking derivatives with respect to the parameters, we get the maximum likelihood estimation of π^q_i, β^2, γ^2, ρ_i and σ^2. Furthermore, π^d_m in Equation (7.3) can be easily estimated during training by counting a histogram of pitch transition patterns.

One issue associated with parameter estimation is that certain pitch values or pitch transitions may not exist in the training set. To compensate for this problem, we apply Dirichlet smoothing [187] to obtain robust estimates of prior probabilities of both pitch and pitch transition. For the latter case, we have

$$\tilde{\pi}_m^d = \frac{\pi_m^d + \lambda}{1 + M\lambda}. \tag{7.7}$$

The choice of λ depends on the amount of training data available. By choosing a very large λ, the prior will be close to a uniform distribution. In the case of pitch priors, we treat the voiced and unvoiced states separately.

$$\tilde{\pi}_i^q(\text{new}) = \begin{cases} \dfrac{N-1}{N} \dfrac{\pi_i^q/(1-\pi_N^q) + \lambda}{1 + (N-1)\lambda} & i = 1..N-1 \\ \dfrac{1}{N} & i = N \end{cases} \tag{7.8}$$

In practice, we choose a relative small λ for π_m^d such that $\tilde{\pi}_m^d \approx \pi_m^d$, and choose a relatively large λ for π_i^q such that $\tilde{\pi}_i^q$ is close to a uniform distribution (this is because π_i^q are usually biased due to the number of training set speakers is very limited).

With regard to decoding, we use the graph in Figure 7.1. We define the forward probability as $\alpha_t(j) = P(Q_t = j, O_{1:t}^q, O_{2:t}^d, R_{2:t})$, and the forward pass is derived as follows,

$$\begin{aligned} \alpha_1(j) &= \pi_j^q P(O_1^q = \phi_t | Q_1 = j); \\ \alpha_t(j) &= \pi_j^q P(O_t^q = \phi_t | Q_t = j) \cdot \sum_i \Bigg[\sum_m \pi_m^d P(O_t^d = \Delta e_t | D_t = m) \cdot \\ & \quad P(D_t = m | Q_{t-1} = i, Q_t = j) \alpha_{t-1}(i) \Bigg] \end{aligned} \tag{7.9}$$

Again, $P(D_t = m | Q_{t-1} = i, Q_t = j)$ is an indicator function. The optimal pitch period sequence is obtained via backtracking after the DP terminates. Notice that if we let $C_{t,i} = -\ln \alpha_t(i)$, Equation (7.9) is equivalent to the DP in Equation (7.1), and the Gaussian assumption of local probabilities leads to a quadratic form of the cost functions. With parameters optimized in the maximum likelihood sense, these functions give remarkably good performance as we will see in Section 7.1.4. It is worth noting that these cost functions can take on other forms under a different distribution assumption, and the parameters can be efficiently estimated as long as good sufficient statistics exist for that distribution.

7.1.4 Experiments

This subsection presents pitch estimation experiments on a database with laryngograph waveforms. A Laryngograph monitors and records the opening and closure of vocal folds, and is a relatively reliable measure of the fundamental frequency. Two databases were combined to create training and test sets for our graphical-model based pitch tracker. One is “Mocha-TIMIT,” [188] developed at Queen Margaret University College, and the other was developed at the Hong Kong University of Science and Technology for tone-estimation research, both with laryngograph recordings.

A total of 1192 continuous English speech utterances (440K frames) from two male and two female speakers were allocated to the training set. The test set was comprised of 4 subsets, corresponding to the same four speakers, amounting to 647 utterances (240K frames) different from the training set. To obtain the ground truth of pitch values, we filtered out the humming noise generated by the electronic devices in a laryngograph, then applied ESPS pitch tracking tool “get_f0” [172] to these waveforms.

Our training and decoding were implemented using GMTK. We ran both get_f0 and our graphical-model based pitch tracker on the speech waveforms of the test set, and compared the results with the ground truth generated using get_f0 on the laryngograph waveforms. The pitch trackers were evaluated in two aspects: pitch estimation and voicing decision [189]. Pitch estimation error rate is measured in terms of *gross error rate* (GER), which is the percentage of pitch estimates that deviate from the ground truth by a certain amount (20% in our experiments). The voicing decision is measured in terms of the percentage of both unvoiced-to-voiced and voiced-to-unvoiced errors. As is shown in Table 7.1 and Table 7.2, both pitch estimation GERs and voicing detection error rates of our pitch tracker were lower than those of get_f0 for all four speakers. Moreover, the graphical structures with and without $O_t^d = \Delta e_t$ features are almost not significantly different. The former structure worked slightly better in pitch estimation and the latter slightly better in voicing decision.

7.2 Adaptive Filtering

One challenge faced by our earlier VJ systems [36] is that a mouse movement was constrained to be in a number of (four or eight) principle directions. As shown in Figure 6.1, we categorized vowel qualities and mapped them to a discrete set of directions. Although we were using soft classification,

Table 7.1: Pitch estimation GER; The highlighted entries include the best error rate and those **not** significantly different from the best at the $p < 0.001$ level.

speaker	female 1	female 2	male 1	male 2
get_f0	5.83	2.11	3.73	1.51
GM without Δe	3.61	**1.53**	**1.06**	**0.86**
GM with Δe	**3.38**	**1.55**	1.17	**0.86**

Table 7.2: Voicing decision error rate; The highlighted entries include the best error rate and those **not** significantly different from the best at the $p < 0.001$ level.

speaker	female 1	female 2	male 1	male 2
get_f0	13.07	12.92	25.66	12.84
GM without Δe	**8.58**	**11.81**	**24.57**	**6.00**
GM with Δe	9.12	12.59	**24.37**	**5.94**

which theoretically could produce movements in arbitrary directions if given ”neutral” vowels as input, in practice the chance of having such outputs is very small due to the nature of the classifier (as will be explained soon). Additionally, we have found that it can be difficult for a user to articulate such interpolated vowels in a continuous and precise manner. Therefore, tasks like drawing a smooth curve was beyond the ability of the early VJ system.

This section introduces an adaptive filtering algorithm in the Vocal Joystick setting that utilizes context information and applies real-time inference in a continuous space. Here by “adaptive” we mean that the filter parameters are updated on the fly during actual inference. This should be distinguished from the term “adaptation” in earlier chapters, which is part of a training process. The VJ system using this algorithm is endowed with the ability to produce movements in arbitrary directions and the ability to draw smooth curves. We first formally introduce the problem; then we describe and compare two VJ system configurations and present in detail our proposed algorithm; finally we discuss qualitative tests and provide comments.

7.2.1 Problem formulation

As mentioned in Chapter 6, we want to produce a sequence of relative movements given a vowel articulation. To this end, we utilize a vowel classifier $g(\mathbf{x}_t)$ that produces a vowel posterior probability vector $\mathbf{z}_t$ given input $\mathbf{x}_t$ at frame t. Then, a transformation $\mathbf{v}_t = f(\mathbf{z}_t)$ maps $\mathbf{z}_t$ to a unit motion vector $\mathbf{v}_t$. In the VJ system, we use a linear transformation as shown in Equation (6.5) for simplicity. The goal of the VJ mouse control can be formulated as follows: assuming that a user's *intended* unit motion vector at time t is $\mathbf{v}_t^{int}$, we desire that

$$f(g(\mathbf{x}_t)) = \mathbf{v}_t^{int} \tag{7.10}$$

Several stages in this process, however, can encounter errors or constraints, posing potential challenges in VJ control. The first possible error is due to human imprecision in articulation. As mentioned in the introduction, it is sometimes difficult for a user to precisely make the vowel that will produce his/her intended motion vector. An analogy is when a beginning violinist plays a note on a violin, it is quite likely to be out of tune. Second, the vowel classifier may not be accurate, leading to system errors in classification. The analogous scenario is that the violin itself may be out of tune. More importantly, there are inherent system constraints in the classification process. Since $g(\cdot)$ is usually a nonlinear transformation and $f(\cdot)$ is a linear transformation, some values of $f(g(\mathbf{x}_t))$ will be more likely to occur than others given that $\mathbf{x}_t$ is uniformly distributed. Consequently, the mouse will be more likely to move along certain directions. Taking the violin analogy again, imagine we replace the violin with a piano, we will then lose the ability to produce certain pitches and pitch variations because of the constraints imposed by the pitch-quantized equal-tempered keyboard.

Our design goals for the VJ system are that it should maximally reduce these errors and constraints by considering the following factors: (1) *producibility*, the system should use easily-producible vowels, reducing the effect of human imprecision; (2) *discriminability*, the system should use distinct vowels, reducing the chance of system errors; (3) *flexibility*, the system should provide enough degrees of freedom in direction control; (4) *predictability*, the system should work in a relatively intuitive way; and (5) *cognitive load*, the system should try to minimize the user's cognitive load. There certainly exist tradeoffs between these factors — for example, to increase flexibility, we may want to increase the number of vowel classes, but this may sacrifice producibility, discriminability, and cognitive load. The adaptive filtering algorithm we propose in Subsection 7.2.3 provides a way to balance these tradeoffs.

7.2.2 A natural strategy

A natural strategy associated with the soft classification scheme is to choose a number of vowel classes, *e.g.* four or eight, and map them to directions as in Figure 6.1. Specifically, we let $g(\cdot)$ be a two-layer MLP classifier, which ideally will output posterior probabilities of the classes if trained using the minimum relative entropy objective [145]. We also apply a supervised speaker adaptation algorithm for MLPs to improve classification accuracies (Chapter 5). Furthermore, we apply a linear transformation $f(\cdot)$ as defined in Equation (6.5).

We first chose to use four vowel classes at the corners of the vowel triangle, namely /æ/, /ɑ/, /u/ and /i/, to maximize discriminability. As suggested in the previous section, a significant drawback of this system is the lack of flexibility. Due to the nature of MLP classifiers, the posterior probability of one vowel class is usually much higher than the others. This results in a system that witnesses mouse movements only along four cardinal directions. We therefore call it a “4-way” system.

To increase flexibility, we developed an “8-way” system using all eight vowel classes, namely /æ/, /ɑ/, /a/, /o/, /u/, /ɨ/, /i/ and /e/. The 8-way system relaxes the constraints imposed by the 4-way system to a great extent. For example, if we want to move the cursor along the up-right direction, in the 4-way system we might have to do it in a zig-zag pattern by saying “/æ/-/ɑ/” repeatedly, while in the 8-way system we can simply say “/a/”. The 8-way system, however, is obviously less advantageous compared with the 4-way system in terms of producibility and discriminability. In fact, we found that many users have trouble producing certain vowels, such as /a/ and /ɨ/. Even when a user can produce all eight vowels, it is sometimes hard to distinguish them since they are less separated in vowel space. Frame-level vowel classification experiments in Chapter 5 showed that the 8-way system has a classification error rate of 8.16% (for 2-second data per vowel) while that of the first system is only 0.19%.

The next question is, can we combine the advantages of both systems? In other words, we desire a system that allows mouse movements in more directions but using only four explicit vowel classes. To this end, it is helpful to infer the user’s intended motion vector by incorporating context information. For example, when the user says "/æ/-/ɑ/-/æ/-/ɑ/-..." in a target acquisition task, it is likely that he wants to move the mouse diagonally.

7.2.3 *An online adaptive filter*

There has been research on plan recognition which aims to infer the plans of an intelligent agent from observations of the agent's actions [190]. A recent trend in approaching the plan recognition problem is to first construct a dynamic Bayesian network (DBN) for plan execution and then to apply inference on this model [191, 192]. To model the plan hierarchy, [192] adopts a model of abstract Markov policies that enables an abstract policy to invoke more refined policies. In such techniques, however, user intent is usually modeled in a finite state space, and inference is often achieved via sampling. This poses problems to the situation where user intent is best expressed as a continuous variable.

We introduce an adaptive filter algorithm of inferring intended values $\mathbf{y}_t$ from noisy estimates $\mathbf{z}_t$. Then we replace $\mathbf{z}_t$ with $\mathbf{y}_t$ in Equation (6.5) in an attempt to obtain the intended motion vector $\mathbf{v}_t^{int}$. The model we use is essentially a hierarchical DBN. The idea is to predict the current $\mathbf{y}_t$ by a "plan" variable, a continuous version of the "abstract policy" used in [192], and then update $\mathbf{y}_t$ based on the current measurement. The system dynamics can be modeled in such a way that the standard Kalman filter algorithm [193] is directly applicable, and hence $\mathbf{y}_t$ can be exactly inferred in the maximum likelihood sense. This dynamic model is "adaptive" in the sense that the model parameters are updated on the fly as will be seen.

Specifically, as shown in Figure 7.3, we model the dynamics of the system using two variables in parallel. Variable $\mathbf{y}_t$ represents a user's intent, and variable $\mathbf{g}_t$ represents the long-term trend of $\mathbf{y}_t$ (or the "plan"). In other words, $\mathbf{y}_t$ can be considered a local goal and $\mathbf{g}_t$ a global goal, both of which are not directly observable. We assume that $\mathbf{y}_t$ can be predicted by $\mathbf{g}_{t-1}$ with certain deviation, *i.e.* $\mathbf{y}_t = \mathbf{g}_{t-1} + \mathbf{u}_t$, where $\mathbf{u}_t$ is a defined as a multivariate Gaussian with covariance matrix $Q(t) = \mathrm{E}[\mathbf{u}_t \mathbf{u}_t^T]$ for computational simplicity. This deviation is caused by plan changes, human imprecision, system constraints (e.g. only allowing mouse movements in certain directions) or the user's intentional adjustments to compensate for previous errors. On the other hand, $\mathbf{g}_t$ can be determined by applying a low-pass filter on $\mathbf{y}_t$. The dynamics of $\mathbf{y}_t$ and $\mathbf{g}_t$ are thereby modeled

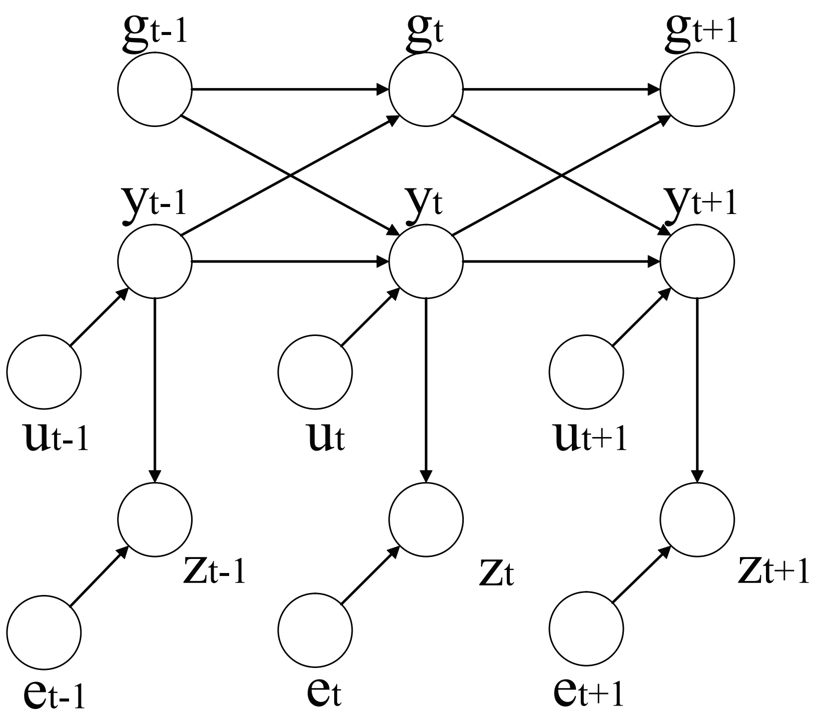


Figure 7.3: A modified Kalman filter for user intent tracking

by linear equations.

$$\begin{bmatrix} \mathbf{g}_{t+1} \\ \mathbf{y}_{t+1} \end{bmatrix} = \begin{bmatrix} a(t)I & (1-a(t))I \\ I & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{g}_t \\ \mathbf{y}_t \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{u}_t \end{bmatrix} \tag{7.11}$$

Furthermore, $\mathbf{z}_t$ is a measurement variable, representing the noisy estimate of $\mathbf{y}_t$. Specifically $\mathbf{z}_t = \mathbf{y}_t + \mathbf{e}_t$, where deviation $\mathbf{e}_t$ is due to system errors or environmental noise. For simplicity, we assume that $\mathbf{e}_t$ is generated from a multivariate Gaussian distribution with covariance matrix $R(t) = \mathrm{E}[\mathbf{e}_t\mathbf{e}_t^T]$,

If we define $\mathbf{x}_t = [\mathbf{g}_t, \mathbf{y}_t]^T$, $\mathbf{w}_t = [0, \mathbf{u}_t]^T$, and $Q'(t) = \mathrm{E}[\mathbf{w}_t\mathbf{w}_t^T]$, we get the standard state-space model. The dynamic and observation models hence become

$$\mathbf{x}_{t+1} = A(t)\mathbf{x}_t + \mathbf{w}_t \tag{7.12}$$

$$\mathbf{z}_t = C\mathbf{x}_t + \mathbf{v}_t \tag{7.13}$$

where

$$\begin{aligned} A(t) &= \begin{bmatrix} a(t)I & (1-a(t))I \\ I & 0 \end{bmatrix} && (7.14) \\ C &= \begin{bmatrix} 0 & I \end{bmatrix} && (7.15) \end{aligned}$$

with $a(t) \in [0, 1]$ being a scalar. In this way, $\mathbf{y}_t$ can be obtained in the maximum likelihood sense using the standard Kalman filter algorithm [193].

$$\hat{\mathbf{x}}_{t+1|t+1} = \hat{\mathbf{x}}_{t+1|t} + K_{t+1}(\mathbf{z}_{t+1} - C^T\hat{\mathbf{x}}_{t+1|t}) \tag{7.16}$$

where

$$\begin{aligned} \hat{\mathbf{x}}_{t+1|t} &= A(t)\hat{\mathbf{x}}_{t|t} \\ K_{t+1} &\triangleq P_{t+1|t}C(C^TP_{t+1|t}C + R(t))^{-1} \\ P_{t+1|t} &= AP_{t|t}A^T + Q'(t)^T \\ P_{t+1|t+1} &= [I - K_{t+1}C^T]P_{t+1|t} \end{aligned} \tag{7.17}$$

It is worth mentioning that we infer $\mathbf{y}_t$ instead of $\mathbf{g}_t$ because $\mathbf{y}_t$ does not have phase delay with respect to $\mathbf{z}_t$ whereas $\mathbf{g}_t$ does, as will be illustrated by a simulation in the next section.

The choice of the model parameters is rather application-specific. First, matrix $A(t)$ decides how stable the plan variable $\mathbf{g}_t$ is. In the extreme case where $a(t) = 1$, $\mathbf{g}_t$ becomes a constant. In *target acquisition* tasks (*e.g.* browsing websites by clicking on hyperlinks), the goal is to get to a target position as quickly as possible starting from the current position. This can be achieved by moving the cursor along the direction that points to the target. In this case, the plan is to move cursor along a fixed direction, and we can use a function $a(t)$ monotonically increasing over time, *e.g.* $a(t) = 1 - c/(t + c)$, so that $\mathbf{g}_t$ converges to a constant as time goes. Note that $A(t)$ is always reset to its initial value once a pause or a stop in articulation is detected. In *steering tasks*, however, the goal is to move the cursor along certain track which can be an arbitrary curve. In such tasks, the plan may change constantly, and we thus let $a(t) = \alpha$, where $\alpha \in [0, 1]$ is empirically chosen to determine the smoothness of the plan or to be determined by a separate acoustic parameter such as pitch.

The covariance matrix $Q(t)$ adjusts the tradeoff between the smoothness of the estimate trajectory and the loyalty to the measurement trajectory. If the variance of $\mathbf{u}_t$ is small, $\mathbf{y}_t$ will converge to its long-term trend $\mathbf{g}_t$ and the trajectory of the estimates becomes smooth; otherwise $\mathbf{y}_t$ will be more loyal to the measurements $\mathbf{z}_t$. This parameter also adjusts the tradeoff between the system's automation degree and the system's predictability to humans. Increased automation management is not always desirable since it yields increased unpredictability to humans, which explains why "even perfect intent inference might not be good enough" [194]. Finally, the covariance matrix $R(t)$ depends on the classifier confusion matrix and the environmental noise condition, both of which can be estimated using principled approaches.

7.2.4 Experiments and Discussions

To illustrate the behavior of the adaptive filter, we ran a simulation for a univariate random process. The measurement $\mathbf{z}_t = sin\frac{\pi}{10}t + \mathbf{e}_t$, where $\mathbf{e}_t \sim \mathcal{N}(0, 0.01)$. The black trajectory in Figure 7.4 represents this noisy sinusoid function for $t = 1 : 100$. Assume that the values +1 and -1 represent two principle directions, and that the oscillating pattern implies the user's effort to find a direction in between, represented by the value 0. We hope that our model can aid the user to approach and stabilize in this desired direction. Here we let $a(t) = 1-1/t$, $\mathrm{E}[\mathbf{u}_t\mathbf{u}_t^T] = 0.1/t$, and $\mathrm{E}[\mathbf{e}_t\mathbf{e}_t^T] = 0.01$. The estimated plan variable $\mathbf{g}_t$ is depicted as the blue trajectory, and $\mathbf{y}_t$ is depicted as the red trajectory in Figure 7.4. The $\mathbf{y}_t$ variable (the red), which is inferred by our algorithm, is loyal to the $\mathbf{z}_t$ (the black) at the beginning and approaches $\mathbf{g}_t$ (the blue) as time goes. This plot also illustrates that $\mathbf{y}_t$ is synchronized with $\mathbf{z}_t$, while $\mathbf{g}_t$ is not.

As a test using real-world applications, the authors used the 4-way system, the 8-way system, and the 4-way system enhanced by adaptive filtering to browse a website. The VJ engine uses a two-layer MLP classifier with 50 hidden units and 7 frames of MFCC features as input. As can be seen in the video demonstration in [195], the adaptive filter manifested more control flexibility while using only four vowels. Using this system, the user achieved fairly stable movements along directions other than the four cardinal ones by oscillating between two vowels. The video also shows the case where the cursor path became a smooth curve when the user transitioned from one vowel to another.

The curve-drawing capability of the adaptive filter is more pronounced in a steering task. This involves using the VJ mouse to steer along two different shapes, a circle and a square tilted by an

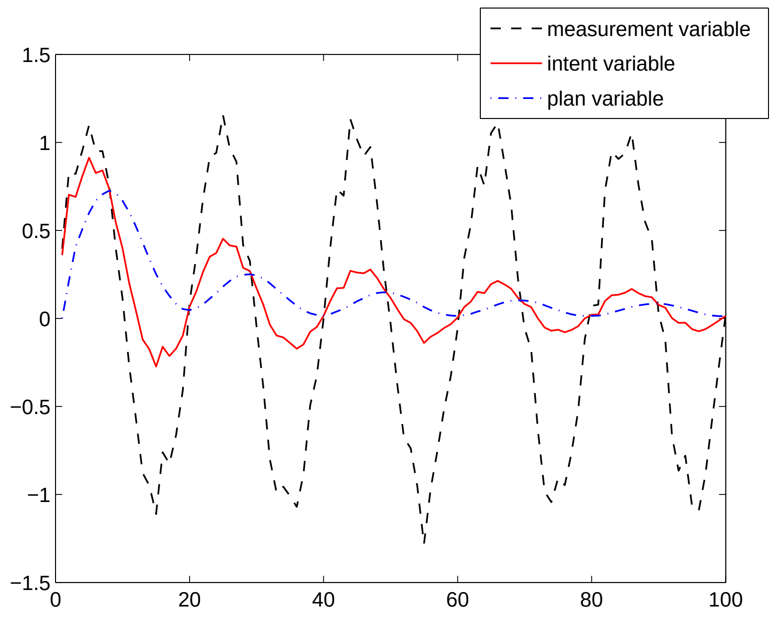
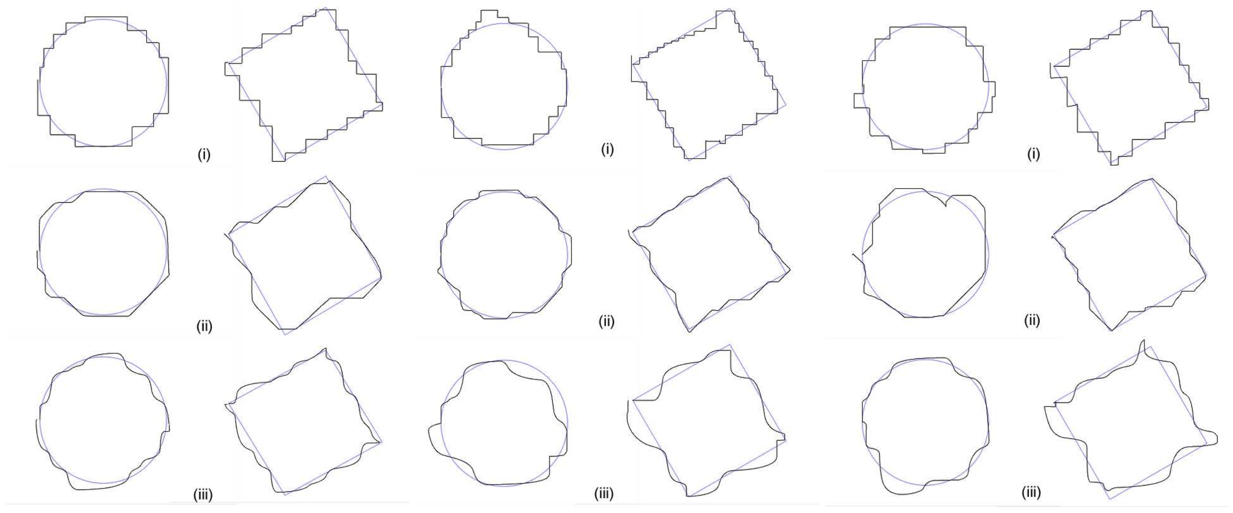


Figure 7.4: Adaptive filtering simulation

angle, shown as the blue paths in Figure 7.2.4. The circle had a radius of 300 pixels on a 800 × 600 screen, and the square was rotated 30 degrees counterclockwise with each side approximately 532 pixels. The cursor always started at the leftmost part of each shape, and its movement was constrained to be within a “tube”, with a radius of 30 pixels, centered at the reference shape. The session would fail once the cursor hit the wall of that tube. The users (again the authors), though having experience with the early VJ system, were relatively novice users of the adaptive filter. As shown in the Figure 7.2.4, the 8-way system and the system with adaptive filtering produced much smoother paths compared with the 4-way system, but the adaptive filter approach achieved this by using only four vowels. Furthermore, the task completion times were very similar across all three systems. The average completion time of tracing the circle is 21-23 seconds for all three systems, while that of tracing the tilted square is 22-28 seconds.

We found in the steering tasks, however, that the adaptive filter enhanced flexibility at the cost of predictability. In other words, the way the system works is not as intuitive as those using the



(a) User I (b) User II (c) User III

Figure 7.5: Steering Snapshots: (i) 4-way system; (ii) 8-way system; (iii) 4-way system with adaptive filtering

natural control strategy; the smoothness of the curve is sometimes hard to control. However, we believe the predictability can be significantly increased if given more time to learn this system — analogous once again to learning to play a violin, individuals with motor impairments, moreover, are often quite motivated to learn novel user interface technologies. Given the experience we had using all three systems, we are encouraged by the prospect of beginning a large-scale user study to thoroughly evaluate user preferences and learnability of these control strategies. In addition, we will consider using a different vocal parameter, such as pitch, to determine the smoothness parameters.

Chapter 8

CONCLUSIONS AND FUTURE WORK

This chapter summarizes the main conclusions of the dissertation, discusses its impact and limitations, and suggests directions for future research.

8.1 Summary of Main Contributions

Adaptation of statistical classifiers is critical when a target (or testing) distribution is different from the distribution that governs training data. In such cases, a classifier optimized for the training distribution needs to be adapted for optimal use in the target distribution. While a vast amount of practical work on adaptation has been done in the areas of speech recognition, natural language processing and pattern recognition, there lacks a unified and principled approach that is applicable to a variety of classifiers, and there lacks a quantitative relationship between the adaptation sample size and the divergence between training and target distributions.

The first contribution of this dissertation is the use of the Bayesian "fidelity prior" that unifies adaptation strategies for a number of different classifiers, as was discussed in Chapter 4. Loosely speaking, this prior measures how likely a classifier is given a training distribution, and directly relates to the KL-divergence between the training and target distributions. In situations where direct optimization is intractable, we replace the fidelity prior with its lower bound in the optimization objective. Specifically, we developed adaptation criteria for a school of generative models including Gaussian models, Gaussian mixture models and hidden Markov models. These criteria resemble the objectives of MAP adaptation [104] but were derived from a different perspective — to minimize the KL-divergence. With regard to discriminative models, we focused our attention on generalized log linear models, and derived a simple adaptation criterion that can be used for adapting SVMs, MLPs and CRFs, while a similar strategy for adapting MaxEnt models has been proposed in [122].

Furthermore, in the PAC-Bayesian setting, we derived several generalization error bounds for adaptation. For a countable function space, we utilized the Occam's Razor bound that bounds the true error by the empirical error plus a capacity term which depends on the prior of the model. Applying the fidelity prior in this setting, the capacity term becomes a monotonically increasing function of the KL-divergence between the training distribution and the distribution determined by the model of interest. An implication of this bound is that unless the KL-divergence between training and target distributions is larger than a threshold (denoted as β in Corollary 4.4.1), it is theoretically better to use the fidelity prior than using the standard prior in learning an adapted model. Next, we derived PAC-Bayesian bounds for Gibbs classifiers which are applicable to both countable and uncountable function spaces, as shown in Corollary 4.4.2 and Corollary 4.4.3. Although we do not yet have a theoretical result to bound the true error for an uncountable function space of point-estimate classifiers, we have empirically shown that fewer samples were needed for smaller KL values to achieve the same confidence.

The second contribution comes from the derivation of regularized adaptation algorithms and applying them to two pattern classification tasks. In Chapter 5 we derived updating formula for adapting GMMs, SVMs and MLPs respectively, and we concentrated on the last two classifiers as they had received relatively less attention from the adaptation perspective. We compared a number of SVM and MLP adaptation algorithms in a vowel classification task and an object recognition task. Our first finding is that without adaptation, MLPs outperformed SVMs in vowel classification, whereas SVMs worked significantly better than MLPs in object recognition. Secondly, an adapted classifier was in general advantageous over its unadapted and retrained counterparts. Finally, when comparing different adaptation techniques, we found that regularized adaptation in general worked very well, as shown in Table 8.1 which summarizes the best SVM and MLP adaptation algorithms in the pattern classification tasks. Additionally, in situations where the amounts of adaptation data were unbalanced across classes, regularized adaptation for MLPs had a pronounced advantage over any other MLP adaptation techniques we were aware of.

Yet another important contribution of this dissertation is the co-development of the Vocal Joystick and the application of regularized adaptation to the VJ system. The VJ is a human-computer interface for individuals with motor impairments. The key feature of the VJ is that it continuously generates acoustic parameters and transforms them into various control parameters. This results in a highly responsive, continuous interaction where a change in vocal characteristics initiated by a user is reflected immediately upon the interface. The contributions of this dissertation to the VJ

Table 8.1: The best-performing adaptation algorithms in two pattern classification tasks.

classifier/task	Vowel classification	Object recognition
SVM	Regularized (5.19); Retrained	Extended regularized (old SV coeffs are updated); Boosted [130]
MLP	Regularized (5.28); Retrained first layer	Regularized (5.28); Retrained speaker-independent

engine are primarily in the signal processing and pattern recognition modules. Chapter 6 described engineering details of voice activity detection, low-level feature extraction, and acoustic parameter estimation which includes pitch tracking, vowel classification, and discrete sound recognition and rejection. While many of these techniques have been well-studied in the literature, applying them to a new problem (the VJ) requires careful considerations and great engineering efforts. The design and implementation of these modules was a trial-and-error process that involved the developer's own experience with the VJ as well as feedback from a number of user studies [36, 37, 49, 53, 54]. The greatest impact of this work to the VJ is the development of efficient adaptation techniques, which is in fact the original motivation of this dissertation. The regularized adaptation algorithms for GMMs and MLPs described in Chapter 5 have been successfully integrated into the VJ engine, and have yielded substantial benefits to system performance.

Furthermore, we extended our discussions on the pattern recognition module in Chapter 7, where we presented novel algorithms on pitch tracking and on post-processing of vowel classification outputs. More specifically, we described a graphical model framework to automatically learn pitch tracking parameters. In this framework, probabilistic dependencies between pitch, pitch transition and acoustical observations are expressed using the language of graphical models. Moreover, the introduction of soft evidence has greatly enhanced the modeling power of the graph and hence the pitch estimation performance. Experiments have shown that our algorithm significantly outperformed ”get_f0” (a standard pitch tracking tool) in both pitch estimation and voicing decision. The second algorithm introduced in Chapter 7 is a novel adaptive filter applied to vowel classification outputs. The algorithm enables the VJ to produce movements in arbitrary directions and to draw smooth curves without phase delay. This adaptive filter is essentially a Kalman filter in which a global-intent variable is modeled in parallel to a local-intent variable, and in which model parameters are adapted on-the-fly. Although there has not been a large-scale user study to conclude on user preferences (compared to the 4-way and the 8-way systems), this technique has by no doubt offered an alternative control strategy which is potentially useful in applications such as drawing.

8.2 Future Work

This section discusses a number of limitations of this dissertation work, and suggests directions for future research.

From the theoretical perspective, a major goal of this dissertation is to study the relationship between adaptation error bounds and the divergence between training and target distributions. To this end, we have utilized PAC-Bayesian theorems [60] to relate the error bounds to KL-divergence. The limitation therein is that the Occam's Razor bound is only valid for a countable function space, while McAllester's PAC-Bayesian bounds are concerned with Gibbs classifiers or Bayesian prediction classifiers. In practice, we are often more interested in point-estimate classifiers in an uncountable function space, such as GMMs, SVMs and MLPs, but PAC-Bayesian theorems are not directly applicable to such situations. An alternative approach is to develop such error bounds in the line of VC theorems [23], where the capacity measures, such as VC entropy or VC dimension, are applicable to both countable and uncountable function spaces. This dissertation has made an initial attempt by deriving the VC dimension of linear classifiers in a constrained function space (for adaptation), but it is somewhat difficult to relate the error bound to the divergence between training and target distributions. One way to circumvent this problem is to use other distortion measures between training and target domains rather than the KL-divergence, as proposed in [196]. Another direction is to study the problem from the multi-task learning point of view [26, 28, 30, 31, 35], as discussed in the introduction. Future work should investigate these avenues.

Algorithm-wise, while regularized adaptation is a principled and simple framework that is amenable to variations in the amount of adaptation data, it still depends on cross-validation to discover the best tradeoff coefficients. It would be interesting to quantify the relationship between the accuracy-regularization frontier [197] (or the regularization path) and the adaptation sample size. Moreover, we have found that adapting the input-to-hidden layer of an MLP gives surprisingly good performance compared to adapting only the hidden-to-output layer, but we have not yet related this regularization strategy to the fidelity prior — only the regularization of the hidden-to-output layer is directly derived from the fidelity prior. Also, this method gives rise to a potential strategy for adapting SVMs, *i.e.*, to adapt kernel parameters or the kernel itself, since the kernel function of an SVM is analogous to the input-to-hidden layer of an MLP. Kernel adaptation can be potentially achieved by exploiting kernel learning techniques such as [147].

With regard to the Vocal Joystick, there are a couple of components in the pattern recognition module that deserve further research. First, the pitch tracker should ideally be trained on a corpus that matches the target application. For example, the VJ may require more drastic pitch changes in certain applications, but such pitch transition patterns are rarely seen in speech databases. On the other hand, it would be expensive and time-consuming to collect a ”VJ-pitch” corpus with laryngograph waveforms. A more realistic solution would be to fine-tune the graphical model parameters empirically in the context of a VJ application. Second, our user-independent discrete sound recognizer was trained on TIMIT, a continuous speech corpus, which is again inconsistent with our target application. Although adaptation can mitigate the mismatch problem, the recognition and rejection performance is more or less affected by the poor unadapted model. An effective way to improve the discrete sound recognizer, therefore, is to collect a ”VJ-discrete-sound” corpus and to retrain the HMMs accordingly. Moreover, the discrete sound vocabulary plays an important role in the rejection mechanism, the design of which has been a tradeoff between avoiding confusion with vowels (for continuous control) and avoiding confusion with garbage sounds such as breathing and spike noise. The current vocabulary, by using unvoiced consonants only, has maximally reduced false positives caused by vowels, but is somewhat vulnerable to environmental noise. It is worth revisiting the problem and designing a vocabulary that balances this tradeoff. Additionally, exploiting information external to the VJ engine is likely to help enhance the performance of the pattern recognition module. For example, given the layout of a computer screen and the current mouse pointer position, it would be easier to predict what the next action would be; but this has to be done in a more controlled environment.

Finally, we would like to note that the problem addressed by this dissertation is one single aspect of adaptive learning. There are many other aspects that have drawn increasing attention in machine learning, *e.g.*, semi-supervised and unsupervised adaptation [10, 14, 15, 198], feature-based adaptation [125, 126] and multi-task learning [26–30]. While the time scale of a Ph.D. is too short to explore all these problems, I am excited at the prospect of continuing research on adaptive learning in the future.

BIBLIOGRAPHY

[1] J. Paredis, “Learning at the crossroads of biology and computation,” in *Proc. 1st Intl. Symp. on Intelligence in Neural and Biological Systems*, May 1995, pp. 56–63.

[2] J. Berko, “The child’s learning of english morphology,” *Word*, vol. 14, 1958.

[3] S. M. Cormier and J. D. Hagman (Eds.), *Transfer of learning: Contemporary research and applications*, Academic Press, 1987.

[4] R. Clark, “Learning vs. performance: Retention and transfer of knowledge and skills from long-term memory,” in *Building Expertise, Cognitive Methods for Training and Performance Improvement*, 1998.

[5] Z. Ghahramani, “Unsupervised learning,” in *Advanced Lectures on Machine Learning*, O. Bousquet, G. Rätsch, and U. von Luxburg, Eds. 2004, vol. 3176 of *Lecture Notes in Computer Science*, Springer.

[6] B. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London, 1986.

[7] L. Kaufman and P. Rousseeuw, *Finding groups in Data: an introduction to cluster analysis*, Wiley, New York, 1990.

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

[9] A. Hyv arinen and E. Oja, “Independent component analysis: algorithms and applications,” *Neural Networks*, vol. 13, 2000.

[10] X. Zhu, “Semi-supervised learning literature survey,” Tech. Rep., Computer Sciences, University of Wisconsin-Madison, 2006.

[11] D. Miller and H. Uyar, “A mixture of experts classifier with learning based on both labeled and unlabeled data,” in *Advances in Neural Information Processing Systems*, 1996.

[12] V. Roth and V. Steinhage, “Nonlinear discriminant analysis using kernel functions,” in *Advances in Neural Information Processing Systems*, 1999.

[13] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, “Text classification from labeled and unlabeled documents using em,” *Machine Learning*, vol. 39, pp. 103134, 2000.

[14] M. R. Amini and P. Gallinari, “Semi-supervised logistic regression,” in *15th European Conference on Artificial Intelligence*, 2002.

[15] Y. Grandvalet and Y. Bengio, “Semi-supervised learning by entropy minimization,” in *Advances in Neural Information Processing Systems*, 2004.

[16] A. Blum and T. Mitchell, “Combining labeled and unlabeled data with co-training,” in *COLT: Proceedings of the Workshop on Computational Learning Theory*, 1998.

[17] K. Nigam and R. Ghani, “Analyzing the effectiveness and applicability of co-training,” in *Ninth International Conference on Information and Knowledge Management*, 2000.

[18] A. Blum and S. Chawla, “Learning from labeled and unlabeled data using graph mincut,” in *Proc. Intl. Conf. on Machine Learning*, 2001.

[19] D. Zhou, O. Bousquet, J. Weston T. N. Lal, and B. Schlkopf, “Learning with local and global consistency,” in *NIPS*, 2003.

[20] X. Zhu, Z. Ghahramani, and J. Lafferty, “Semi-supervised learning using gaussian fields and harmonic functions,” in *Proc. Intl. Conf. on Machine Learning*, 2003.

[21] D. Angluin, “Queries revisited,” *Theoretical Computer Science*, vol. 313, no. 2, 2004.

[22] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby, “Selective sampling using the query by committee algorithm,” *Machine Learning*, vol. 28, pp. 133–168, 1997.

[23] V. N. Vapnik, *Statistical Learning Theory*, Wiley-Interscience, 1998.

[24] K. Bennett, “Combining support vector and mathematical programming methods for classification,” in *Advances in kernel methods - support vector learning*, B. Schökopf et. al., Ed. 1999, MIT-Press.

[25] T. Joachims, “Transductive inference for text classification using support vector machines,” in *Proc. Intl. Conf. on Machine Learning*, 1999.

[26] J. Baxter, “Learning internal representations,” in *COLT: Proceedings of the Workshop on Computational Learning Theory*. 1995, Morgan Kaufman.

[27] J. Baxter, “A model of inductive bias learning,” *Journal of Artificial Intelligence Research*, vol. 12, pp. 149–198, 2000.

[28] S. Thrun and L.Y. Pratt, *Learning To Learn*, Kluwer Academic Publishers, Boston, MA, 1998.

[29] S. Ben-David and R. Schuller, “Exploiting task relatedness for multiple task learning,” in *Proceedings of Computational Learning Theory (COLT)*, 2003.

[30] R. K. Ando and T. Zhang, “A framework for learning predictive structures from multiple tasks and unlabeled data,” *Journal of Machine Learning Research*, vol. 6, 2005.

[31] R. Caruana, “Multitask learning,” *Machine Learning Journal*, vol. 28, 1997.

[32] T. Evgeniou and M. Pontil, “Regularized multi-task learning,” in *SIGKDD*, August 2004.

[33] J. Baxter, “A Bayesian/information theoretic model of learning to learn via multiple task sampling,” *Machine Learning*, 1997.

[34] T. Heskes, “Empirical Bayes for learning to learn,” in *Proc. Intl. Conf. on Machine Learning*, 2000.

[35] R. Raina, A. Y. Ng, and D. Koller, “Constructing informative priors using transfer learning,” in *Proc. Intl. Conf. on Machine Learning*, 2006.

[36] J. Bilmes et. al., “The vocal joystick: A voice-based human-computer interface for individuals with motor impairments,” in *Human Language Technology Conf. and Conf. on Empirical Methods in Natural Language Processing*, 2005.

[37] J. Bilmes and et.al., “The Vocal Joystick,” in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, May 2006.

[38] G. Faconti and M. Massink, “Continuity in human computer interaction,” Tech. Rep., CHI Workshop, 2000.

[39] “Origin instruments,” 2007, http://www.orin.com.

[40] “Pride mobility products corp,” 2007, http://www.pridemobility.com.

[41] R. J. K. Jacob, “What you look at is what you get: eye movement-based interaction techniques,” in *Proc. of the SIGCHI conf. on Human Factors in Computing Systems*, 1990.

[42] X.Huang and et. al., “Mipad: a multimodal interaction prototype,” in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, May 2001.

[43] B. Manaris and A. Harkreader, “Suitekeys: A speech understanding interface for the motor-control challenged,” in *In Proc. ACM SIGCAPH Conf on Assistive Technologies*, 1998.

[44] T.Igarashi, “Voice as sound: using non-verbal voice input for interactive control,” in *In Proc. ACM Symp. on User Interface Software and Technology*, 2001.

[45] Y. Mihara, E. Shibayama, and S. Takahashi, "The migratory cursor: accurate speech-based cursor movement by moving multiple ghost cursors using non-verbal vocalizations," in *Proc. of the 7th intl. ACM conference on Computers and accessibility*, 2005.

[46] C. de Mauro, M. Gori, M. Maggini, and E. Martinelli, "A voice device with an application-adapted protocol for microsoft Windows," in *Proc. Intl. Conf. on Multimedia Computing and Systems*, 1999.

[47] A. S. Karimullah and A. Sears, "Speech-based cursor control," in *Proceedings of Assets 2002*. 2002, pp. 178–185, ACM Press.

[48] X. Li and J. Bilmes, "A Bayesian divergence prior for classifier adaptation," in *Eleventh Intl. Conf. on Artificial Intelligence and Statistics*, 2007.

[49] X. Li and J. Bilmes, "Regularized adaptation of discriminative classifiers," in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, September 2006.

[50] X.Li, J.Bilmes, and J.Malkin, "Maximum margin learning and adaptation of MLP classifiers," in *Eurospeech*, September 2005.

[51] X. Li, J. Malkin, and J. Bilmes, "Graphical model approach to pitch tracking," in *Proc. Intl. Conf. on Spoken Language Processing*, Oct 2004.

[52] X. Li, J. Malkin, S. Harada, J. Bilmes, R. Wright, and J. Landay, "An online adaptive filtering algorithm for the Vocal Joystick," in *Interspeech*, September 2006.

[53] J. Malkin, X. Li, and J. Bilmes, "Energy and loudness for speed control in the Vocal Joystick," in *ASRU Workshop*, Nov 2005.

[54] S. Harada, J. Landay, J. Malkin, X. Li, and Jeff Bilmes, "The Vocal Joystick: Evaluation of voice-based cursor control techniques," in *Proc. of the 8th Intl. ACM SIGACCESS Conf. on Computers and Accessibility*, October 2006.

[55] K. Kilanski, J. Malkin, X. Li, R. Wright, and J. Bilmes, "The vocal joystick data collection effort and vowel corpus," in *Interspeech*, 2006.

[56] S. Arora, L. Babai, J. Stern, and Z. Sweedyk, "The hardness of approximate optima in lattices, codes, and systems of linear equations," *Journal of Computer and System Sciences*, 1997.

[57] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *Journal of the American Statistical Association, 101*, 2006.

[58] M. Anthony and P. L. Bartlett, *Neural Network Learning: Theoretical Foundation*, Cambridge University Press, 1999.

[59] G. Lugosi and K. Zeger, “Concept learning using complexity regularization,” *IEEE Transactions on Information Theory*, vol. 42, no. 1, 1996.

[60] D. A. McAllester, “PAC-Bayesian stochastic model selection,” *Machine Learning Journal*, 2001.

[61] B. Schölkopf and A. J. Smola, *Learning with kernels*, The MIT Press, 2001.

[62] M.Jordan and C.Bishop, *An Introduction to Graphical Models*, pre-print, 2001.

[63] A.P. Dempster, N.M. Laird, and D.B. Rubin, “Maximum-likelihood from incomplete data via the EM algorithm,” *J. Royal Statist. Soc. Ser. B.*, vol. 39, 1977.

[64] R. J. A. Little and D. B. Rubin, *Statistical Analysis with Missing Data*, Wiley, New York, 1987.

[65] L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer, “Maximum mutual information estimation of HMM parameters for speech recognition,” in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, December 1986, pp. 49–52.

[66] B.-H. Juang, W. Hou, and C.-H. Lee, “Minimum classification error rate methods for speech recognition,” *IEEE Trans. on Speech and Audio Processing*, vol. 5, pp. 257–265, 1997.

[67] Y. Ephraim and L. Rabiner, “On the relation between modeling approaches for speech recognition,” *IEEE Trans. on Information Theory*, vol. 36, no. 2, 1990.

[68] E. McDermott, T.J. Hazen, J. Le Roux, A. Nakamura, and S. Katagiri, “Discriminative training for large vocabulary speech recognition using minimum classification error,” *IEEE Trans. on Audio, Speech and Language Processing*, Jan 2007.

[69] B. Taskar, *Learning structured prediction models: a large margin approach*, Ph.D. thesis, Stanford University, December 2004.

[70] F. Sha and L. Saul, “Large margin hidden Markov models for automatic speech recognition,” in *Advances in Neural Information Processing Systems*, 2006.

[71] J. Bilmes, “Dynamic Bayesian Multinets,” in *Proceedings of the 16th conf. on Uncertainty in Artificial Intelligence*. 2000, Morgan Kaufmann.

[72] J. Bilmes, G. Zweig, T. Richardson, K. Filali, K. Livescu, P. Xu, K. Jackson, Y. Brandman, E. Sandness, E. Holtz, J. Torres, and B. Byrne, "Discriminatively structured graphical models for speech recognition: JHU-WS-2001 final workshop report," Tech. Rep., CLSP, Johns Hopkins University, 2001.

[73] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Advances in Neural Information Processing Systems*, 1998.

[74] H. Ney, "On the probabilistic interpretation of neural network classifiers and discriminative training criteria," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 2, 1995.

[75] P. C. Woodland and D. Povey, "Large scale discriminative training of hidden markov models for speech recognition," *Computer Speech and Language*, 2002.

[76] J. C. Spall, *Introduction to Stochastic Search and Optimization*, Wiley, 2003.

[77] P. S. Gopalakrishnan, D. Kanevsky, A. Nadas, and D. Nahamoo, "An inequality for rational functions with applications to some statistical estimation problems," *IEEE Trans. on Information Theory*, vol. 37, pp. 107–113, 1991.

[78] C.Cortes and V.Vapnik, "Support vector networks," *Machine Learning*, 1995.

[79] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of online learning and an application to boosting," *Jounral of Computer and System Sciences*, pp. 119–139, 1997.

[80] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: a new explanation for the effectiveness of voting methods," in *Proc. 14th Intl. Conf. on Machine Learning*, 1997.

[81] G. Lugosi and N. Vayatis, "On the Bayes risk consistency of regularized boosting methods," *Annals of Statistics*, 2003.

[82] T. Zhang, "Statistical behavior and consistency of classification methods based on convex risk minimization," *Annals of Statistics*, 2003.

[83] D. A. McAllester, "Some PAC-bayesian theorems," in *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1998.

[84] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, Wiley, 1991.

[85] J. Langford, "Tutorial on practical prediction theory for classification," *Journal of Machine Learning Research*, pp. 273–306, March 2005.

[86] L. G. Valiant, "A theory of the learnable," *Communications of ACM*, vol. 27, no. 11, 1984.

[87] M. Seeger, "The proof of McAllester's PAC-Bayesian theorem," in *Advances in Neural Information Processing Systems*, 2002.

[88] J. Langford, M. Seeger, and N. Megiddo, "An improved predictive accuracy bound for averaging classifiers," in *Proc. 18th Intl. Conf. on Machine Learning*, 2001, pp. 290–297.

[89] R. Herbrich and T. Graepel, "A PAC-Bayesian margin bound for linear classiers; why SVMs work," in *Advances in Neural Information Processing Systems*, 2001.

[90] R. Meir and T. Zhang, "Generalization error bounds for Bayesian mixture algorithms," *Journal of Machine Learning Research*, vol. 4, no. 5, pp. 839–860, 2003.

[91] T. Jaakkola, M. Meila, and T. Jebara, "Maximum entropy discrimination," Tech. Rep. AITR-1668, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1999.

[92] J. Langford and J. Shawe-Taylor, "PAC-Bayes and margins," in *Advances in Neural Information Processing Systems*, 2002.

[93] Y. Singer and M. K. Warmuth, "Training algorithms for hidden markov models using entropy based distance functions," in *Advances in Neural Information Processing Systems*, 1997, vol. 9, p. 641.

[94] J. Cohen, T. Kamm, and A. Andreou, "Vocal tract normalization in speech recognition: compensating for systematic speaker variability," *Journal of Acoustic Society of America*, vol. 97, no. 5, 1995.

[95] L.Welling and H.Ney, "Speaker adaptive modeling by vocal tract normalization," *IEEE Trans. on Speech and Audio Processing*, vol. 10, no. 6, 2002.

[96] M. S. Bartlett, *Face Image Analysis by Unsupervised Learning*, Kluwer Academic Publishers, Massachusetts, 2001.

[97] T. Riklin-Raviv and A. Shashua, "The quotient image: Class based recognition and synthesis under varying illumination," in *CVPR*, 1999.

[98] R. Ramamoorthi, "Analytic pca construction for theoretical analysis of lighting variability in images of a lambertian object," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[99] C. Leggetter and P. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Computer, Speech and Language*, vol. 9, 1995.

[100] M.Gales and P.Woodland, "Mean and variance adaptation within the mllr framework," *Computer, Speech and Language*, vol. 10, 1996.

[101] M. Gales and P. Woodland, "Variance compensation within the MLLR framework," Tech. Rep. CUED/FINFENG/TR.242, Cambridge Univ., 1996.

[102] M. Gales, "The generation and use of regression class trees for mllr adaptation," Tech. Rep. CUED/FINFENG/TR.263, Cambridge Univ., 1996.

[103] J.-L. Gauvain and C.-H. Lee, "Bayesian learning of Gaussian mixture densities for hidden Markov models," in *Proceedings of the DARPA Speech and Natural Language Workshop*. 1991, pp. 272–277, Morgan Kaufmann.

[104] J.-L.Gauvain and C.-H.Lee, "Maximum a posteriori estimation for multivariate gaussian mixture observations of Markov chains," *IEEE Trans. on Speech and Audio Processing*, vol. 2, 1994.

[105] G.Zavaliagkos, R.Schwarz, J.McDonogh, and J.Makhoul, "Adaptation algorithms for large scale HMM recognizers," in *Proc. Eurospeech*, 1995.

[106] O. Siohan, C. Chesta, and C. Lee, "Hidden markov model adaptation using maximum a posteriori linear regression," in *Workshop on Robust Methods for Speech Recognition in Adverse Conditions*, 1999.

[107] C. Chesta, O. Siohan, and C. Lee, "Maximum a posteriori linear regression for hidden Markov model adaptation," in *Eurospeech*, 1999.

[108] T.-A. Myrvoll, O. Siohan, C.-H. Lee, and W. Chou, "Structural maximum a posteriori linear regression for unsupervised speaker adaptation," in *Proc. Intl. Conf. on Spoken Language Processing*, 2000.

[109] A. Surendran and C.-H. Lee, "Transformation based bayesian prediction to adaptaion of hmms," *Speech Communications*, vol. 34, pp. 159–174, 2001.

[110] J.-T. Chien, "Linear regression based Bayesian predictive classification for speech recognition," *IEEE Trans. on Speech and Audio Processing*, vol. 11, no. 1, 2003.

[111] K. Yu and M. Gales, "Incremental adaptation using Bayesian inference," in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, 2006.

[112] T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul, "A compact model for speaker-adaptive training," in *Proc. Intl. Conf. on Spoken Language Processing*, Philadelphia, PA, 1996, pp. 1137–1140.

[113] M. Padmanabhan, L. R. Bahl, D. Nahamoo, and M. A. Picheny, "Speaker clustering and transformation for speaker adaptation in speech recognition systems," *IEEE Trans. on Speech and Audio Processing*, pp. 71–77, 1998.

[114] M. Gales, "Cluster adaptive training of hidden markov models," *IEEE Trans. on Speech and Audio Processing*, pp. 417–428, 2000.

[115] R.Kuhn, J.-C.Junqua, P.Nguyen, and N.Niedzielski, "Rapid speaker adaptation in eigenvoice space," *IEEE Trans. on Speech and Audio Processing*, vol. 8, 2000.

[116] J. Kwok, B. Mak, and S. Ho, "Eigenvoice speaker adaptation via composite kernel pca," in *Advances in Neural Information Processing Systems*, 2004.

[117] V. Dounipiotis and Y.-G. Deng, "Eigenspace-based MLLR with speaker adaptive training in large vocabulary conversational speech recognition," in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, 2004.

[118] B. Mak and R. Hsiao, "Kernel eigenspace-based mllr adaptation using multiple regression classes," in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, 2005.

[119] Hal Daumé III and Daniel Marcu, "Domain adaptation for statistical classifiers," *Journal of Articial Intelligence Research 26*, pp. 1–15, 2006.

[120] M. Bacchiani and B. Roark, "Unsupervised langauge model adaptation," in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, 2003.

[121] S. Ross, *Introduction to Probability Models, the Eighth Edition*, Elsevier, 2003.

[122] C. Chelba and A. Acero, "Adaptation of maximum entropy capitalizer: Little data can help a lot," in *Empirical Methods in Natural Language Processing*, July 2004.

[123] P. R. Clarkson and A. J. Robinson, "Language model adaptation using mixtures and an exponentially decaying cache," in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, 1997.

[124] T. Jebara and A. Pentland, "Maximum conditional likelihood via bound maximization and the cem algorithm," in *Advances in Neural Information Processing Systems*, 1998.

[125] R. Florian and et. al., "A statistical model for multilingual entity detection and tracking," in *NAACL/HLT*, 2004.

[126] J. Blitzer, R. McDonald, and F. Pereira, "Domain adaptation with structural correspondence learning," in *Empirical Methods in Natural Language Processing*, 2006.

[127] J. Stadermann and G. Rigoll, “Two-stage speaker adaptation of hybrid tied-posterior acoustic models,” in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, 2005.

[128] J.Neto, L. Almeida, M. Hochberg, C. Martins, L. Nunes, S. Renals, and T. Robinson, “Speaker-adaptation for hybrid HMM-ANN continuous speech recognition system,” in *Proc. Eurospeech*, 1995.

[129] V. Abrash, H. Franco, A. Sankar, and M. Cohen, “Connectionist speaker normalization and adaptation,” in *eurospeech*, 1995.

[130] N. Matic, I. Guyon, J. Denker, and V. Vapnik, “Writer adaptation for on-line handwritten character recognition,” in *Proc. Intl. Conf. on Pattern Recognition and Document Analysis*, 1993.

[131] B.-B.Peng, Z.-X.Sun, and X.-G. Xu, “SVM-based incremental active learning for user adaptation for online graphics recognition system,” in *Proc.Intl.Conf.on Machine Learning and Cybernetics*, 2002.

[132] S.Rüping, “Incremental learning with support vector machines,” in *Proc. IEEE. Intl. Conference on Data Mining*, 2001.

[133] P. Wu and T. G. Dietterich, “Improving svm accuracy by training on auxiliary data sources,” in *Proc. Intl. Conf. on Machine Learning*, 2004.

[134] M. Sugiyama and K.-R. Mller, “Input-dependent estimation of generalization error under covariate shift,” *Statistics & Decisions*, vol. 23, no. 4, 2005.

[135] Y. Singer and M.K. Warmuth, “Training algorithm for hidden markov models using entropy based distance functions,” in *Advances in Neural Information Processing System*, 1996.

[136] M. N. Do, “Fast approximation of Kullback-Leibler distance for dependence trees and hidden Markov models,” *IEEE Signal Processing Letters*, vol. 10, 2003.

[137] N. Vasconcelos, “On the efficient evaluation of probabilistic similarity functions for image retrieval,” *IEEE Transactions on Information Theory*, vol. 50, no. 7, 2004.

[138] J. Silva and S. Narayanan, “Upper bound kullback-leibler divergence for hidden markov models with application as discrimination measure for speech recognition,” in *Intl. Symposium on Information Theory*, July 2006.

[139] J. Platt, “Probabilistic outputs for support vector machines and comparison to regularized likelihood methods,” in *Advances in Large Margin Classifiers*, A.J. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, Eds., 2000, pp. 61–74.

[140] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2006.

[141] E. Parrado-Hernandez A. Ambroladze and J. Shawe-Taylor, "Learning the prior for the PAC-Bayes bound," Tech. Rep., Southampton, UK, 2004.

[142] J. Bilmes, "A gentle tutorial on the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," Tech. Rep. ICSI-TR-97-021, University of California, Berkeley, 1997.

[143] R.Collobert and S.Bengio, "Links between perceptrons, MLPs and SVMs," in *Intl. Conf. on Machine Learning*, 2004.

[144] J. Goodman, "Exponential priors for maximum entropy models," in *HLT/NAACL*, 2003.

[145] C. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.

[146] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[147] C.P. Diehl and G. Cauwenberghs, "Svm incremental learning, adaptation and optimization," in *Proc. IEEE Int. Joint Conf. Neural Networks (IJCNN'2003)*, 2003.

[148] T. J. Hastie and R. J. Tibshirani, "Classification by pairwise coupling," in *Advances in Neural Information Processing Systems*, 1998.

[149] J. Weston and C. Watkins, "Multi-class support vector machines," Tech. Rep. CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, 1998.

[150] Y. Le Cun, B. Boser, J. S Denker, D. Henderson, R. E.. Howard, W. Howard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems II (Denver 1989)*, D. S. Touretzky, Ed., pp. 396–404. Morgan Kaufmann, San Mateo, CA, 1990.

[151] S. Rosset, J. Zhu, and T. Hastie, "Margin maximizing loss functions," in *Advances in Neural Information Processing Systems*, 2004.

[152] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. 18th International Conf. on Machine Learning*. 2001, pp. 282–289, Morgan Kaufmann, San Francisco, CA.

[153] A. L. Berger, S. Della Pietra, and V. J. Della Pietra, "A maximum entropy approach to natural language processing," *Computational Linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[154] S. Della Pietra, V. J. Della Pietra, and J. D. Lafferty, “Inducing features of random fields,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 380–393, 1997.

[155] P. Jantke, “Types of incremental learning,” in *AAAI Symposium on Training Issues in Incremental Learning*, 1993.

[156] N. Syed, H. Liu, and K. Sung, “Incremental learning with support vector machines,” in *Proc. Workshop on Support Vector Machines at the Intl. Joint Conf. on Aritifical Intelligence*, 1999.

[157] Gert Cauwenberghs and Tomaso Poggio, “Incremental and decremental support vector machine learning,” in *Advances in Neural Information Processing Systems*, 2000, pp. 409–415.

[158] R.Collobert and S.Bengio, “SVMTorch: support vector machines for large-scale regression problems,” *The journal of Machine Learning Research*, 2001.

[159] J. Platt, “Sequential minimal optimization: A fast algorithm for training support vector machines,” Tech. Rep. 98-14, Microsoft Research, Redmond, 1998.

[160] H.Bourlard and N.Morgan, *Connectionist Speech Recognition: A Hybrid Approach*, Kluwer Academic Publishers, 1994.

[161] D. Burton, *On the inverse shortest path problem*, Ph.D. thesis, Department of Mathematics, FUNDP, Namur, Belgium, 1993.

[162] J. Zhang and Z. Ma, “Solution structure of some inverse combinatorial optimization problems,” *Journal of Combinatorial Optimization*, vol. 3, no. 1, 1999.

[163] C. Heuberger, “Inverse optimization: A survey on problems, methods, and results,” *Journal of Combinatorial Optimization*, vol. 8, no. 3, pp. 329–361, 2004.

[164] R. K. Ahuja and J. B. Orlin, “Inverse optimization,” *Operations Research*, vol. 49, no. 5, 2001.

[165] V.W.Zue, S.Seneff, and J.Glass, “Speech database development at MIT: TIMIT and beyond,” *Speech Communication*, vol. 9, no. 4, pp. 351–356, 1990.

[166] Y. LeCun, F. J. Huang, and L. Bottou, “Learning methods for generic object recognition with invariance to pose and lighting,” in *Computer Vision and Pattern Recognition*, 2004.

[167] T.W.Stewart and N. Vaillette, *Language Files*, The Ohio State University Press, 2001.

[168] “International phonetic alphabet,” http://www.arts.gla.ac.uk/ipa/ipa.html.

[169] J.R.Deller, J.H.L.Hansen, and J.G.Proakis, *Discrete-time processing of speech signals*, Macmillan, 1993.

[170] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing*, Prentice Hall, 2001.

[171] L.R.Rabiner, “On the use of autocorrelation analysis for pitch detection,” *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 25, 1977.

[172] D. Talkin, *Speech coding and synthesis*, Elsevier Science B.V, 1995.

[173] S. Young, G. Evermann, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK book 3.2*, Cambrideg University and Microsoft Corporation, 2000’s.

[174] S. R. Young, “Detecting misrecognitions and out-ofvocabulary words,” in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, 1994.

[175] M. G. Rahim, C. H. Lee, and B.-H. Juang, “Robust utterance verification for connected digits recognition,” in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, 1995.

[176] Z. Rivlin, M. Cohen, V. Abrash, and T. Chung, “A phone-dependent confidence measure for utterance rejection,” in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, 1996.

[177] J. G. A. Dolfing and A. Wendemuth, “Combination of confidence measures in isolated word recognition,” in *Proc. Intl. Conf. on Spoken Language Processing*, 1998.

[178] W. Hess, *Pitch Determination of Speech Signals*, Springer-Verlag, 1983.

[179] B.S.Atal, “Automatic speaker recognition based on pitch contours,” *Journal of the Acoustical Society of America*, vol. 52, no. 6, 1972.

[180] N.Kunieda, T.Shimamura, and J.Suzuki, “Robust method of measurement of fundamental frequency by ACOLS-autocorrelation of log spectrum,” in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, 1996.

[181] D-J.Liu and C-T.Lin, “Fundamental frequency estimation based on the joint time-frequency analysis of harmonic spectral structure,” *IEEE Trans. on Speech and Audio Processing*, vol. 9, no. 6, 2001.

[182] A.Cheveigne and H.Kawahara, “YIN, a fundamental frequency estimator for speech and music,” *Journal of the Acoustical Society of America*, vol. 111, no. 4, 2002.

[183] J.Droppo and A.Acero, "Maximum a posteriori pitch tracking," in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, 1998.

[184] J. Bilmes and G. Zweig, "The Graphical Models Toolkit: An open source software system for speech and time-series processing," in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, 2002.

[185] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of P lausible Inference*, Morgan Kaufmann, 2nd printing edition, 1988.

[186] J. Bilmes, "On soft evidence in Bayesian networks," Tech. Rep. UWEETR-2004-0016, Dept. of Electrical Engineering, University of Washington, 2004.

[187] I.J.Good, *The estimation of probabilities: an essay on modern Bayesian methods*, MIT Press, Cambridge, MA, 1965.

[188] A. Wrench, "A multichannel/multispeaker articulatory database for continuous speech recognition research," in *Workshop on Phonetics and Phonology in ASR*, 2000.

[189] L.R.Rabiner, M.J.Cheng, and A.E.Rosenberg, "A comparative performance study of several pitch detection algorithms," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 24, 1976.

[190] H. Kautz and J. F. Allen, "Generalized plan recognition," in *AAAI*, 1986, pp. 32–38.

[191] D. Pynadath and M. Wellman, "Accounting for context in plan recognition, with application to traffic monitoring," in *Uncertainty in Artificial Intelligence*, 1995.

[192] H. H. Bui, S. Venkatesh S, and G. West, "The recognition of abstract Markov policies," in *AAAI*, 2000, pp. 524–530.

[193] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice*, Prentice Hall, 1993.

[194] H. B. Funk and C. A. Miller, "User acceptance and plan recognition: Why even perfect intent inferencing might not be good enough," in *AAAI Fall Symposium*, 2001.

[195] "The Vocal Joystick," 2007, http://ssli.ee.washington.edu/vj.

[196] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of representations for domain adaptation," in *Advances in Neural Information Processing Systems 20*, Cambridge, MA, 2007, MIT Press.

[197] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu, “The entire regularization path for the support vector machine,” *Journal of Machine Learning Research*, vol. 5, pp. 1391–1415, Oct 2004.

[198] K. Shinoda and C.-H. Lee, “Unsupervised adaptation using structural Bayes approach,” in *Proc. IEEE Intl. Conf. on Acoustic, Speech and Signal Processing*, 1998.

[199] R. Fletcher, *Practical Methods of Optimization, 2nd Edition*, Wiley, 2000.

Appendix A
PROOFS

A.1 Proof of Theorem 2.3.3

Proof Since $Q(\cdot) = [0, 1]$, according to Hoeffding inequality,

$$\begin{aligned} & \Pr\big(R(f) - R_{\text{emp}}(f) \ge \epsilon\big) \le \exp(-2m\epsilon^2) \overset{\Delta}{=} \delta \\ \Rightarrow \quad & \Pr\left(R(f) \ge R_{\text{emp}}(f) + \sqrt{-\frac{\log\delta}{2m}}\right) \le \delta \end{aligned}$$

Then using the above and union bound theorem, we have

$$\begin{aligned} & \Pr\left(\forall i, R(f_i) \le R_{\text{emp}}(f_i) + \sqrt{-\frac{\log(\pi(f_i)\delta)}{2m}}\right) \\ = \quad & 1 - \Pr\left(\exists i, R(f_i) \ge R_{\text{emp}}(f_i) + \sqrt{-\frac{\log(\pi(f_i)\delta)}{2m}}\right) \\ \ge \quad & 1 - \textstyle\sum_i \Pr\left(R(f_i) \ge R_{\text{emp}}(f_i) + \sqrt{-\frac{\log(\pi(f_i)\delta)}{2m}}\right) \\ \ge \quad & 1 - \textstyle\sum_i \pi(f_i)\delta = 1 - \delta \quad \blacksquare \end{aligned}$$

A.2 Proof of Equation (2.22)

Proof To prove this, we first show that for any $(\mathbf{x}, y)$, if sgn $f_{Bayes}(\mathbf{x}) =$ sgn $\big(\mathrm{E}_{q(f)}[f(\mathbf{x})]\big) \neq y$, then $\mathrm{E}_{q(f)}[\mathrm{I}(f(\mathbf{x}) \neq y)] \ge 1/2$. In particular, for $y = 1$ and a fixed $\mathbf{x}$,

$$\begin{aligned} \text{sgn }\big(\mathrm{E}_{q(f)}[f(\mathbf{x})]\big) \neq y \quad & \Longleftrightarrow \mathrm{E}_{q(f)}[f(\mathbf{x})] < 0 \\ & \Longleftrightarrow \textstyle\int_{f:f(\mathbf{x})=-1} q(f)\cdot(-1)\,df + \int_{f:f(\mathbf{x})=1} q(f)\cdot 1\,df < 0 \\ & \Longleftrightarrow \textstyle\int q(f)\,\mathrm{I}(f(\mathbf{x})=1)\,df < \int (f)\,\mathrm{I}(f(\mathbf{x})=-1)\,df \end{aligned}$$

Since $\int_{f:f(\mathbf{x})=1} q(f)\,df + \int_{f:f(\mathbf{x})=-1} q(f)\,df = 1$, we have

$$\mathrm{E}_{q(f)}[\mathrm{I}(f(\mathbf{x}) \neq 1)] = \int_f q(f)\,\mathrm{I}(f(\mathbf{x}) = -1)\,df \ge 1/2$$

Similary, the same result can be obtained for $y = -1$. Taking expectation w.r.t. $(\mathbf{x}, y)$, we have

$$R_{p(\mathbf{x},y)}(f_{Bayes}) \le 2\,\mathrm{E}_{q(f)}[R_{p(\mathbf{x},y)}(f)] \quad \blacksquare$$

A.3 Proof of Theorem 2.4.1

Here we provide the proof by Schölkopf [61] as this is relevant to the proof of Corollary 4.4.4

Proof Assume that canonical hyperplanes with the constraint $\|\mathbf{w}\| \leq \Lambda$ has a VC-dimension h. In other words, there exists h points $\{\mathbf{x}_i\}_{i=1}^h$, such that for any possible label assignments $\{y_i\}_{i=1}^h$ there always exists a $\mathbf{w}$ with $\|\mathbf{w}\| \leq \Lambda$ such that

$$y_i \langle \mathbf{w}, \mathbf{x} \rangle \geq 1 \; i = 1..h \tag{A.1}$$

We say $\{\mathbf{x}_i\}_{i=1}^h$ are shattered by $\mathbf{w}$. The proof is then two-folds: (1) we show that for any possible label assignments $\{y_i\}_{i=1}^h$, we have $\|\sum_{i=1}^h y_i \mathbf{x}_i\| \leq h/\Lambda$; and (2) we show that there exists a label assignments $\{y_i\}_{i=1}^h$ such that $\|\sum_{i=1}^h y_i \mathbf{x}_i\|^2 \leq hR^2$, where $R = \max_i \|\mathbf{x}_i\|$.

Summing the inequality (A.1) over $i = 1..h$, we have

$$h \leq \langle \mathbf{w}, \sum_{i=1}^h y_i \mathbf{x}_i \rangle \leq \|\mathbf{w}\| \|\sum_{i=1}^h y_i \mathbf{x}_i\| \leq \Lambda \|\sum_{i=1}^h y_i \mathbf{x}_i\|,$$

which proves (1). On the other hand, considering an i.i.d. label assignment of y_i where $p(y_i = 1) = p(y_i = -1) = 1/2$, we have

$$\mathrm{E}_{p(y_{1:h})} \left[\|\sum_{i=1}^h y_i \mathbf{x}_i\|^2 \right] = \sum_{i=1}^h \mathrm{E}_{p(y_{1:h})} \left\langle y_i \mathbf{x}_i, \sum_j y_j \mathbf{x}_j \right\rangle = \sum_{i=1}^h \mathrm{E}_{p(y_{1:h})} \|y_i \mathbf{x}_i\|^2 \leq hR^2$$

Thus there must exsit a label assignments $\{y_i\}_{i=1}^h$ such that $\|\sum_{i=1}^h y_i \mathbf{x}_i\|^2 \leq hR^2$, which proves (2).

Combining (1) and (2), the theorem is proved. ▮

A.4 Proof of Corollary 4.4.4

The theorem can be proved simply by using $\|\mathbf{w}\| = \|\mathbf{w} - \mathbf{w}^{tr} + \mathbf{w}^{tr}\| \leq c + \|\mathbf{w}^{tr}\|$ in Proof A.3.

A.5 Proof of Corollary 4.4.5

Proof Since $y_i \langle \mathbf{w}, \mathbf{x} \rangle \geq 1$, $i = 1..h$, we replace step (1) in Proof A.3 by the following. Summing the above inequality over $i = 1, ..., h$, we have

$$h \leq \langle \mathbf{w} - \mathbf{w}^{tr}, \sum_{i=1}^h y_i \mathbf{x}_i \rangle \leq \|\mathbf{w} - \mathbf{w}^{tr}\| \|\sum_{i=1}^h y_i \mathbf{x}_i\| \leq c \|\sum_{i=1}^h y_i \mathbf{x}_i\|$$

On the other hand, according to step (2) in Proof A.3 , there exists $\{y_i\}_{i=1}^h$, such that

$$\|\sum_{i=1}^{h} y_i \mathbf{x}_i\|^2 \leq hR^2$$

Therefore, $h \leq \Lambda^2 R^2$. ▌

A.6 Proof of Lemma 4.3.2

Proof Let $\hat{\mathbf{x}}$ denote an locally minimum solution to $J_1(\mathbf{x})$. First, we show that the lemma holds true when $\hat{\mathbf{x}} = \mathbf{0}$. For any $\mathbf{x} \neq \hat{\mathbf{x}}$, we have

$$\begin{aligned} & g(0) + \frac{\lambda'}{2}\|0\| < g(\mathbf{x}) + \frac{\lambda'}{2}\|\mathbf{x}\| && \text{(A.2)} \\ \Longrightarrow \quad & g(0) < g(\mathbf{x}) && \text{(A.3)} \\ \Longrightarrow \quad & g(0) + \frac{\lambda}{2}\|0\| < g(\mathbf{x}) + \frac{\lambda}{2}\|\mathbf{x}\| && \text{(A.4)} \end{aligned}$$

Therefore, 0 is also a locally minimum solution to $J_2(\mathbf{x})$.

Secondly, if $\hat{\mathbf{x}} \neq 0$, we study the solution space $\{\mathbf{x} : \mathbf{x} \neq 0\}$ in which $\|\mathbf{x}\|$ becomes twice differentiable. Then we can take the first and second derivatives of J_1. The **necessary conditions** for $\hat{\mathbf{x}}$ to be a locally optimal solution to J_1 is that [199],

$$\begin{aligned} \frac{d}{d\mathbf{x}}\left(g(\mathbf{x}) + \frac{\lambda'}{2}\|\mathbf{x}\|\right)\Big|_{\mathbf{x}=\hat{\mathbf{x}}} &= \frac{d}{d\mathbf{x}}g(\mathbf{x})\Big|_{\mathbf{x}=\hat{\mathbf{x}}} + \frac{\lambda'}{2}\frac{\hat{\mathbf{x}}}{\|\hat{\mathbf{x}}\|} = 0 && \text{(A.5)} \\ \frac{d^2}{d\mathbf{x}^2}\left(g(\mathbf{x}) + \frac{\lambda'}{2}\|\mathbf{x}\|\right)\Big|_{\mathbf{x}=\hat{\mathbf{x}}} &= \frac{d^2}{d\mathbf{x}^2}g(\mathbf{x})\Big|_{\mathbf{x}=\hat{\mathbf{x}}} \quad \text{is positive semidefinite} && \text{(A.6)} \end{aligned}$$

which follow that for $\mathbf{x} \neq 0$, $\frac{d}{d\mathbf{x}}\|\mathbf{x}\| = \frac{d}{d\mathbf{x}}\sqrt{\mathbf{x}^T\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$, and $\frac{d^2}{d\mathbf{x}^2}\|\mathbf{x}\| = 0$. On the other hand, the **sufficient condition** [199] for $\hat{\mathbf{x}}$ to be the optimal solution to J_2 is

$$\begin{aligned} \frac{d}{d\mathbf{x}}\left(g(\mathbf{x}) + \frac{\lambda}{2}\|\mathbf{x}\|^2\right)\Big|_{\mathbf{x}=\hat{\mathbf{x}}} &= \frac{d}{d\mathbf{x}}g(\mathbf{x})\Big|_{\mathbf{x}=\hat{\mathbf{x}}} + \lambda\hat{\mathbf{x}} = 0 && \text{(A.7)} \\ \frac{d^2}{d\mathbf{x}^2}\left(g(\mathbf{x}) + \frac{\lambda}{2}\|\mathbf{x}\|^2\right)\Big|_{\mathbf{x}=\hat{\mathbf{x}}} &= \frac{d^2}{d\mathbf{x}^2}g(\mathbf{x})\Big|_{\mathbf{x}=\hat{\mathbf{x}}} + I \quad \text{is positive definite} && \text{(A.8)} \end{aligned}$$

It is easy to see that (A.5) and (A.6) implies (A.7) and (A.8) if $\lambda = \frac{\lambda'}{2\|\hat{\mathbf{x}}\|}$. Such a choice of λ exists for any λ'. ▌

Appendix B
ALGORITHMS

B.1 EM updating equations for GMM adaptation

Our goal is minimize $R_{\text{emp}}(f) - \lambda \ln p_{\text{fid}}(f)$ whose upper bound is given by Equation (5.4) and (5.5). Now we derive the update equations for ω_y, $c_{y,k}$, $\mu_{y,k}$ and $\Sigma_{y,k}$ respectively. First, we extract out only the terms which depend on ω_y:

$$J(\omega_y) = -\frac{1}{m}\sum_{i=1}^{m} \delta_{i,y} \ln \omega_y - \lambda \omega_y^{tr} \ln \omega_y \tag{B.1}$$

where $\sum_y \omega_y = 1$. By introducing the Lagrangian form $\sum_y J(\omega_y) - \alpha(\sum_y \omega_y - 1)$, it is easy to obtain the optimal solution as

$$\hat{\omega}_y = \frac{\frac{1}{m}\sum_{i=1}^{m} \delta_{i,y} + \lambda \omega_y^{tr}}{1+\lambda}$$

Similarly, we minimize the terms which depend on $c_{y,k}$,

$$J(c_{y,k}) = -\frac{1}{m}\sum_{i=1}^{m} \delta_{i,y} L_{k|y}(i) \ln c_{y,k} - \lambda \omega_y^{tr} c_{y,k}^{tr} \ln c_{y,k} \tag{B.2}$$

and obtain the optimal solution as

$$\hat{c}_{y,k} = \frac{\frac{1}{m}\sum_{i=1}^{m} \delta_{i,y} L_{k|y}(i) + \lambda \omega_y^{tr} c_{y,k}^{tr}}{\frac{1}{m}\sum_{i=1}^{m} \delta_{i,y} + \lambda \omega_y^{tr}}$$

Next, we inspect the terms which depend on $\mu_{y,k}$,

$$\begin{aligned} J(\mu_{y,k}) &= -\frac{1}{m}\sum_{i=1}^{m} \delta_{i,y} L_{k|y}(i) \ln \mathcal{N}(\mathbf{x}_i; \mu_{y,k}, \Sigma_{y,k}) + \\ &\quad \lambda \omega_y^{tr} c_{y,k}^{tr} D(\mathcal{N}(\mathbf{x}; \mu_{y,k}^{tr}, \Sigma_{y,k}^{tr}) || \mathcal{N}(\mathbf{x}_i; \mu_{y,k}, \Sigma_{y,k})) \\ &= \sum_{i=1}^{m} \delta_{i,y} L_{k|y}(i) \frac{1}{2}(\mathbf{x}_i - \mu_{y,k})^T \Sigma_{y,k}^{-1} (\mathbf{x}_i - \mu_{y,k}) + \\ &\quad \lambda \omega_y^{tr} c_{y,k}^{tr} \frac{1}{2}(\mu_{y,k}^{tr} - \mu_{y,k})^T \Sigma_{y,k}^{-1} (\mu_{y,k}^{tr} - \mu_{y,k}) \end{aligned} \tag{B.3}$$

where the second term follows Equation (2.31). Taking the derivative of $J(\mu_{y,k})$ and setting it to zero, we obtain the optimal solution as

$$\hat{\mu}_{y,k} = \frac{\frac{1}{m}\sum_{i=1}^{m}\delta_{i,y}L_{k|y}(i)\mathbf{x}_i + \lambda\omega_y^{tr}c_{y,k}^{tr}\mu_{y,k}^{tr}}{\frac{1}{m}\sum_{i=1}^{m}\delta_{i,y}L_{k|y}(i) + \lambda\omega_y^{tr}c_{y,k}^{tr}}$$

Finally, we inspect the terms which depend on $\Sigma_{y,k}^{-1}$

$$\begin{aligned} J(\Sigma_{y,k}^{-1}) &= \frac{1}{m}\sum_{i=1}^{m}\delta_{i,y}L_{k|y}(i)\Big(-\frac{1}{2}\ln|\Sigma_{y,k}^{-1}| + \frac{1}{2}(\mathbf{x}_i - \mu_{y,k})^T\Sigma_{y,k}^{-1}(\mathbf{x}_i - \mu_{y,k})\Big) + \\ &\lambda\omega_y^{tr}c_{y,k}^{tr}\Big(-\frac{1}{2}\ln|\Sigma_{y,k}^{-1}| + \frac{1}{2}\operatorname{tr}\left(\Sigma_{y,k}^{tr}\Sigma_{y,k}^{-1}\right) + \frac{1}{2}(\mu_{y,k}^{tr} - \mu_{y,k})^T\Sigma_{y,k}^{-1}(\mu_{y,k}^{tr} - \mu_{y,k})\Big) \\ &= \frac{1}{m}\sum_{i=1}^{m}\delta_{i,y}L_{k|y}(i)\Big(-\frac{1}{2}\ln|\Sigma_{y,k}^{-1}| + \frac{1}{2}\operatorname{tr}\left(\Sigma_{y,k}^{-1}M_{i,y,k}\right)\Big) + \\ &\lambda\omega_y^{tr}c_{y,k}^{tr}\Big(-\frac{1}{2}\ln|\Sigma_{y,k}^{-1}| + \frac{1}{2}\operatorname{tr}\left(\Sigma_{y,k}^{tr}\Sigma_{y,k}^{-1}\right) + \frac{1}{2}\operatorname{tr}\left(\Sigma_{y,k}^{-1}N_{y,k}\right)\Big) \end{aligned} \tag{B.4}$$

where $M_{i,y,k} \triangleq (\mathbf{x}_i - \mu_{y,k})(\mathbf{x}_i - \mu_{y,k})^T$ and $N_{y,k} \triangleq (\mu_{y,k}^{tr} - \mu_{y,k})(\mu_{y,k}^{tr} - \mu_{y,k})^T$. Using the fact that for symmetrical matrices A and B,

$$\frac{\partial \ln|A|}{\partial A} = 2A^{-1} - \text{diag}(A^{-1}) \tag{B.5}$$

$$\frac{\text{tr}(AB)}{\partial A} = 2B - \text{diag}(B) \tag{B.6}$$

we can derive the derivative of $J(\Sigma_{y,k}^{-1})$ w.r.t. $\Sigma_{y,k}^{-1}$ as follows

$$\frac{\partial J(\Sigma_{y,k}^{-1})}{\partial \Sigma_{y,k}^{-1}} = \frac{1}{m}\sum_{i=1}^{m}\delta_{i,y}L_{k|y}(i)\Big(-\Sigma_{y,k} + \frac{1}{2}\text{diag}\left(\Sigma_{y,k}\right) + M_{i,y,k} - \frac{1}{2}\text{diag}(M_{i,y,k})\Big) \tag{B.7}$$

$$+\lambda\omega_y^{tr}c_{y,k}^{tr}\Big(-\Sigma_{y,k} + \frac{1}{2}\text{diag}(\Sigma_{y,k}) + \Sigma_{y,k}^{tr} - \frac{1}{2}\text{diag}(\Sigma_{y,k}^{tr}) + N_{y,k} - \frac{1}{2}\text{diag}(N_{y,k})\Big) = 0 \tag{B.8}$$

Furthermore, since $A - \frac{1}{2}\text{diag}(A) = 0$ implies $A = 0$, we obtain the optimal solution as

$$\hat{\Sigma}_{y,k} = \frac{\frac{1}{m}\sum_{i=1}^{m}\delta_{i,y}L_{k|y}(i)M_{i,y,k} + \lambda\omega_y^{tr}c_{y,k}^{tr}\Big(\Sigma_{y,k}^{tr} + N_{y,k}\Big)}{\frac{1}{m}\sum_{i=1}^{m}\delta_{i,y}L_{k|y}(i) + \lambda\omega_y^{tr}c_{y,k}^{tr}}$$

B.2 Sequential minimal optimization for SVM adaptation

We optimize two parameters α_1 and α_2 at a time.

$$\begin{aligned} J(\alpha_1,\alpha_2) = \alpha_1 + \alpha_2 - \tfrac{1}{2}K_{11}\alpha_1^2 - \tfrac{1}{2}K_{22}\alpha_2^2 - sK_{12}\alpha_1\alpha_2 \\ -y_1\alpha_1(v_1+u_1) - y_2\alpha_2(v_2+u_2) + const \end{aligned} \tag{B.9}$$

where $s = y_1 y_2$ and

$$\begin{aligned} v_j &= f(\mathbf{x}_j) - b - y_1\alpha_1 K_{1j} - y_2\alpha_2 K_{2,i} \\ u_j &= \hat{f}(\mathbf{x}_j) - \hat{b} \end{aligned}$$

In SMO, each step finds the maximum in the subspace $\alpha_1 + s\alpha_2 = \gamma$. We replace α_1 by $\gamma - s\alpha_2$,

$$\begin{aligned} J(\alpha_2) = \gamma - s\alpha_2 + \alpha_2 - \tfrac{1}{2}K_{11}(\gamma - s\alpha_2)^2 - \tfrac{1}{2}K_{22}\alpha_2^2 - sK_{12}(\gamma - s\alpha_2)\alpha_2 \\ -y_1(\gamma - s\alpha_2)(v_1+u_1) - y_2\alpha_2(v_2+u_2) + const \end{aligned} \tag{B.10}$$

Taking derivative with respect to α_2 and setting it to zero, we have

$$\alpha_2^{new}(K_{11}+K_{22}-2K_{12}) = s(K_{11}-K_{22})(\alpha_1+s\alpha_2) + y_2(v_1+u_1-v_2-u_2) + 1 - s \tag{B.11}$$

This is the maximum when $\eta \triangleq K_{11}+K_{22}-2K_{12} > 0$.

Finally, we get the core update equation: if $\eta > 0$,

$$\begin{aligned} \alpha_2^{new} &= \alpha_2 + \frac{g(\mathbf{x}_1) - g(\mathbf{x}_2)}{\eta} \\ \alpha_1^{new} &= \gamma - s\alpha_2^{new} \end{aligned} \tag{B.12}$$

where $g(\mathbf{x}_j) = y_1(\hat{f}(\mathbf{x}_1) + f(\mathbf{x}_1)) - 1$. When $\eta > 0$, we apply the standard SMO procedure.

B.3 Stochastic gradient descent for MLP adaptation

Specifically, we consider an MLP with J layers, each with N^{j-1} input nodes and N^j output nodes, where $n_0 = D$ is the dimension of input features. Each layer j, $j = 1..J-1$, performs an affine transformation followed by a sigmoid operation, except that the very last layer J performs an affine followed by a softmax operation. We further define a set of symbols regarding sample i.

First, it is easy to see that

$$\frac{\partial J_i}{\partial \mathbf{w}_n^j} = \frac{\partial J_i}{\partial a_{i,n}^j} \cdot \phi_i^j \quad n = 1..N^j. \tag{B.13}$$

ϕ_i^j: the input vector of layer j;

$\mathbf{a}_i^j$: $\mathbf{a}_i = \mathbf{W}^j \phi_i^j + \mathbf{b}^j$; *i.e.*, $a_{i,n}^j = \langle \mathbf{w}_n^j, \phi_i^j \rangle + b_n^j$, $n = 1..N^j$;

$\mathbf{z}_i$: output vector of layer J, $z_{i,n} = \dfrac{\exp\{-a_{i,n}^J\}}{\sum_{k=1}^{N^J} \exp\{-a_{i,k}^J\}}$, $n = 1..N^J$;

$\mathbf{t}_i$: target label vector, where $t_{i,n} = \mathrm{I}(n = y_i)$

Based on these derivatives , the inner loop of the back-propagation is written as: for $i = 1..m$,

$$\mathbf{w}_n^j = \mathbf{w}_n^j - \eta \left(\frac{\partial J_i}{\partial a_{i,n}^j} \cdot \phi_i^j + \lambda^j (\mathbf{w}_n^j - \mathbf{w}_n^{j\,tr}) \right) \tag{B.14}$$

$$b_n^j = b_n^j - \eta \frac{\partial J_i}{\partial a_{i,n}^j} \tag{B.15}$$

where η is the learning rate.

Now the problem is reduced to deriving $\dfrac{\partial J_i}{\partial a_{i,n}^j}$ for $j = J$ and $j = 1..J-1$ respectively. When $j = J$, we have obtained in Chapter 5 that

$$\frac{\partial J_i}{\partial a_{i,n}^J} = z_{i,n} - t_{i,n} \quad n = 1..N^J. \tag{B.16}$$

When $j = 1..J-1$, we have

$$\frac{\partial J_i}{\partial a_{i,n}^j} = \sum_{k=1}^{N^j} \frac{\partial J_i}{\partial a_{i,k}^{j+1}} \cdot \frac{\partial a_{i,k}^{j+1}}{\partial a_{i,n}^j} \tag{B.17}$$

$$= \sum_{k=1}^{K} \frac{\partial J_i}{\partial a_{i,k}^{j+1}} w_{k,n}^{j+1} \phi_{i,n}^j (1 - \phi_{i,n}^j) \tag{B.18}$$