COMPUTATIONAL IMAGING AND VISION

Machine Learning in Computer Vision

N. Sebe, Ira Cohen, Ashutosh Garg
and Thomas S. Huang

Springer

Machine Learning in Computer Vision

by

N. SEBE

University of Amsterdam,
The Netherlands

IRA COHEN

HP Research Labs, U.S.A.

ASHUTOSH GARG

Google Inc., U.S.A.

and

THOMAS S. HUANG

University of Illinois at Urbana-Champaign,
Urbana, IL, U.S.A.

Springer

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN-10 1-4020-3274-9 (HB) Springer Dordrecht, Berlin, Heidelberg, New York
ISBN-10 1-4020-3275-7 (e-book) Springer Dordrecht, Berlin, Heidelberg, New York
ISBN-13 978-1-4020-3274-5 (HB) Springer Dordrecht, Berlin, Heidelberg, New York
ISBN-13 978-1-4020-3275-2 (e-book) Springer Dordrecht, Berlin, Heidelberg, New York

Published by Springer,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

Printed on acid-free paper

Printed in the Netherlands.

To my parents
Nicu

To Merav and Yonatan
Ira

To my parents
Asutosh

To my students:
Past, present, and future
Tom

Contents

Foreword

It started with *image processing* in the sixties. Back then, it took ages to digitize a Landsat image and then process it with a mainframe computer. Processing was inspired on the achievements of signal processing and was still very much oriented towards programming.

In the seventies, *image analysis* spun off combining image measurement with statistical pattern recognition. Slowly, computational methods detached themselves from the sensor and the goal to become more generally applicable.

In the eighties, model-driven *computer vision* originated when artificial intelligence and geometric modelling came together with image analysis components. The emphasis was on precise analysis with little or no interaction, still very much an art evaluated by visual appeal. The main bottleneck was in the amount of data using an average of 5 to 50 pictures to illustrate the point.

At the beginning of the nineties, vision became available to many with the advent of sufficiently fast PCs. The Internet revealed the interest of the general public im images, eventually introducing *content-based image retrieval*. Combining independent (informal) archives, as the web is, urges for interactive evaluation of approximate results and hence weak algorithms and their combination in weak classifiers.

In the new century, the last analog bastion was taken. In a few years, sensors have become all digital. Archives will soon follow. As a consequence of this change in the basic conditions datasets will overflow. Computer vision will spin off a new branch to be called something like *archive-based* or *semantic vision* including a role for formal knowledge description in an ontology equipped with detectors. An alternative view is *experience-based* or *cognitive vision*. This is mostly a data-driven view on vision and includes the elementary laws of image formation.

This book comes right on time. The general trend is easy to see. The methods of computation went from dedicated to one specific task to more generally applicable building blocks, from detailed attention to one aspect like filtering to a broad variety of topics, from a detailed model design evaluated against a few data to abstract rules tuned to a robust application.

From the source to consumption, images are now all digital. Very soon, archives will be overflowing. This is slightly worrying as it will raise the level of expectations about the accessibility of the pictorial content to a level compatible with what humans can achieve.

There is only one realistic chance to respond. From the trend displayed above, it is best to identify basic laws and then to learn the specifics of the model from a larger dataset. Rather than excluding interaction in the evaluation of the result, it is better to perceive interaction as a valuable source of instant learning for the algorithm.

This book builds on that insight: that the key element in the current revolution is the use of machine learning to capture the variations in visual appearance, rather than having the designer of the model accomplish this. As a bonus, models learned from large datasets are likely to be more robust and more realistic than the brittle all-design models.

This book recognizes that machine learning for computer vision is distinctively different from plain machine learning. Loads of data, spatial coherence, and the large variety of appearances, make computer vision a special challenge for the machine learning algorithms. Hence, the book does not waste itself on the complete spectrum of machine learning algorithms. Rather, this book is focussed on machine learning for pictures.

It is amazing so early in a new field that a book appears which connects theory to algorithms and through them to convincing applications.

The authors met one another at Urbana-Champaign and then dispersed over the world, apart from Thomas Huang who has been there forever. This book will surely be with us for quite some time to come.

Arnold Smeulders
University of Amsterdam
The Netherlands
October, 2004

Preface

The goal of computer vision research is to provide computers with human-like perception capabilities so that they can sense the environment, understand the sensed data, take appropriate actions, and learn from this experience in order to enhance future performance. The field has evolved from the application of classical pattern recognition and image processing methods to advanced techniques in image understanding like model-based and knowledge-based vision.

In recent years, there has been an increased demand for computer vision systems to address "real-world" problems. However, much of our current models and methodologies do not seem to scale out of limited "toy" domains. Therefore, the current state-of-the-art in computer vision needs significant advancements to deal with real-world applications, such as navigation, target recognition, manufacturing, photo interpretation, remote sensing, etc. It is widely understood that many of these applications require vision algorithms and systems to work under partial occlusion, possibly under high clutter, low contrast, and changing environmental conditions. This requires that the vision techniques should be robust and flexible to optimize performance in a given scenario.

The field of machine learning is driven by the idea that computer algorithms and systems can improve their own performance with time. Machine learning has evolved from the relatively "knowledge-free" general purpose learning system, the "perceptron" [Rosenblatt, 1958], and decision-theoretic approaches for learning [Blockeel and De Raedt, 1998], to symbolic learning of high-level knowledge [Michalski et al., 1986], artificial neural networks [Rowley et al., 1998a], and genetic algorithms [DeJong, 1988]. With the recent advances in hardware and software, a variety of practical applications of the machine learning research is emerging [Segre, 1992].

Vision provides interesting and challenging problems and a rich environment to advance the state-of-the art in machine learning. Machine learning technology has a strong potential to contribute to the development of flexible and robust vision algorithms, thus improving the performance of practical vision systems. Learning-based vision systems are expected to provide a higher level of competence and greater generality. Learning may allow us to use the experience gained in creating a vision system for one application domain to a vision system for another domain by developing systems that acquire and maintain knowledge. We claim that learning represents the next challenging frontier for computer vision research.

More specifically, machine learning offers effective methods for computer vision for automating the model/concept acquisition and updating processes, adapting task parameters and representations, and using experience for generating, verifying, and modifying hypotheses. Expanding this list of computer vision problems, we find that some of the applications of machine learning in computer vision are: segmentation and feature extraction; learning rules, relations, features, discriminant functions, and evaluation strategies; learning and refining visual models; indexing and recognition strategies; integration of vision modules and task-level learning; learning shape representation and surface reconstruction strategies; self-organizing algorithms for pattern learning; biologically motivated modeling of vision systems that learn; and parameter adaptation, and self-calibration of vision systems. As an eventual goal, machine learning may provide the necessary tools for synthesizing vision algorithms starting from adaptation of control parameters of vision algorithms and systems.

The goal of this book is to address the use of several important machine learning techniques into computer vision applications. An innovative combination of computer vision and machine learning techniques has the promise of advancing the field of computer vision, which will contribute to better understanding of complex real-world applications. There is another benefit of incorporating a learning paradigm in the computational vision framework. To mature the laboratory-grown vision systems into real-world working systems, it is necessary to evaluate the performance characteristics of these systems using a variety of real, calibrated data. Learning offers this evaluation tool, since no learning can take place without appropriate evaluation of the results.

Generally, learning requires large amounts of data and fast computational resources for its practical use. However, all learning does not have to be online. Some of the learning can be done off-line, e.g., optimizing parameters, features, and sensors during training to improve performance. Depending upon the domain of application, the large number of training samples needed for inductive learning techniques may not be available. Thus, learning techniques should be able to work with varying amounts of a priori knowledge and data.

The effective usage of machine learning technology in real-world computer vision problems requires understanding the domain of application, abstraction of a learning problem from a given computer vision task, and the selection of appropriate representations for the learnable (input) and learned (internal) entities of the system. To succeed in selecting the most appropriate machine learning technique(s) for the given computer vision task, an adequate understanding of the different machine learning paradigms is necessary.

A learning system has to clearly demonstrate and answer the questions like what is being learned, how it is learned, what data is used to learn, how to represent what has been learned, how well and how efficient is the learning taking place and what are the evaluation criteria for the task at hand. Experimental details are essential for demonstrating the learning behavior of algorithms and systems. These experiments need to include scientific experimental design methodology for training/testing, parametric studies, and measures of performance improvement with experience. Experiments that exihibit scalability of learning-based vision systems are also very important.

In this book, we address all these important aspects. In each of the chapters, we show how the literature has introduced the techniques into the particular topic area, we present the background theory, discuss comparative experiments made by us, and conclude with comments and recommendations.

Acknowledgments

This book would not have existed without the assistance of Marcelo Cirelo, Larry Chen, Fabio Cozman, Michael Lew, and Dan Roth whose technical contributions are directly reflected within the chapters. We would like to thank Theo Gevers, Nuria Oliver, Arnold Smeulders, and our colleagues from the Intelligent Sensory Information Systems group at University of Amsterdam and the IFP group at University of Illinois at Urbana-Champaign who gave us valuable suggestions and critical comments. Beyond technical contributions, we would like to thank our families for years of patience, support, and encouragement. Furthermore, we are grateful to our departments for providing an excellent scientific environment.

Chapter 1

INTRODUCTION

Computer vision has grown rapidly within the past decade, producing tools that enable the understanding of visual information, especially for scenes with no accompanying structural, administrative, or descriptive text information. The Internet, more specifically the Web, has become a common channel for the transmission of graphical information, thus moving visual information retrieval rapidly from stand-alone workstations and databases into a networked environment.

Practicality has begun to dictate that the indexing of huge collections of images by hand is a task that is both labor intensive and expensive - in many cases more than can be afforded to provide some method of intellectual access to digital image collections. In the world of text retrieval, text "speaks for itself" whereas image analysis requires a combination of high-level concept creation as well as the processing and interpretation of inherent visual features. In the area of intellectual access to visual information, the interplay between human and machine image indexing methods has begun to influence the development of computer vision systems. Research and application by the image understanding (IU) community suggests that the most fruitful approaches to IU involve analysis and learning of the type of information being sought, the domain in which it will be used, and systematic testing to identify optimal methods.

The goal of computer vision research is to provide computers with human-like perception capabilities so that they can sense the environment, understand the sensed data, take appropriate actions, and learn from this experience in order to enhance future performance. The vision field has evolved from the application of classical pattern recognition and image processing techniques to advanced applications of image understanding, model-based vision, knowledge-based vision, and systems that exhibit learning capability. The ability to reason and the ability to learn are the two major capabilities associated with these systems. In recent years, theoretical and practical advances are being made in the field of computer vision and pattern recognition by new techniques and processes of learning, representation, and adaptation. It is probably fair to claim, however, that learning represents the next challenging frontier for computer vision.

1. Research Issues on Learning in Computer Vision

In recent years, there has been a surge of interest in developing machine learning techniques for computer vision based applications. The interest derives from both commercial projects to create working products from computer vision techniques and from a general trend in the computer vision field to incorporate machine learning techniques.

Learning is one of the current frontiers for computer vision research and has been receiving increased attention in recent years. Machine learning technology has strong potential to contribute to:

- the development of flexible and robust vision algorithms that will improve the performance of practical vision systems with a higher level of competence and greater generality, and

- the development of architectures that will speed up system development time and provide better performance.

The goal of improving the performance of computer vision systems has brought new challenges to the field of machine learning, for example, learning from structured descriptions, partial information, incremental learning, focusing attention or learning regions of interests (ROI), learning with many classes, etc. Solving problems in visual domains will result in the development of new, more robust machine learning algorithms that will be able to work in more realistic settings.

From the standpoint of computer vision systems, machine learning can offer effective methods for automating the acquisition of visual models, adapting task parameters and representation, transforming signals to symbols, building trainable image processing systems, focusing attention on target object, and learning when to apply what algorithm in a vision system.

From the standpoint of machine learning systems, computer vision can provide interesting and challenging problems. As examples consider the following: learning models rather than handcrafting them, learning to transfer experience gained in one application domain to another domain, learning from large sets of images with no annotation, designing evaluation criteria for the quality of learning processes in computer vision systems. Many studies in machine learning assume that a careful trainer provides internal representations of the observed environment, thus paying little attention to the problems of perception. Unfortunately, this assumption leads to the development of brittle systems with noisy, excessively detailed, or quite coarse descriptions of the perceived environment.

Esposito and Malerba [Esposito and Malerba, 2001] listed some of the important research issues that have to be dealt with in order to develop successful applications:

- *Can we learn the models used by a computer vision system rather than handcrafting them?*

  In many computer vision applications, handcrafting the visual model of an object is neither easy nor practical. For instance, humans can detect and identify faces in a scene with little or no effort. This skill is quite robust, despite large changes in the visual stimulus. Nevertheless, providing computer vision systems with models of facial landmarks or facial expressions is very difficult [Cohen et al., 2003b]. Even when models have been handcrafted, as in the case of page layout descriptions used by some document image processing systems [Nagy et al., 1992], it has been observed that they limit the use of the system to a specific class of images, which is subject to change in a relatively short time.

- *How is machine learning used in computer vision systems?*

  Machine learning algorithms can be applied in at least two different ways in computer vision systems:

  - to improve perception of the surrounding environment, that is, to improve the transformation of sensed signals into internal representations, and
  - to bridge the gap between the internal representations of the environment and the representation of the knowledge needed by the system to perform its task.

  A possible explanation of the marginal attention given to learning internal representations of the perceived environment is that feature extraction has received very little attention in the machine learning community, because it has been considered application-dependent and research on this issue is not of general interest. The identification of required data and domain knowledge requires the collaboration with a domain expert and is an important step of the process of applying machine learning to real-world problems.

Only recently, the related issues of feature selection and, more generally, data preprocessing have been more systematically investigated in machine learning. Data preprocessing is still considered a step of the knowledge discovery process and is confined to data cleaning, simple data transformations (e.g., summarization), and validation. On the contrary, many studies in computer vision and pattern recognition focused on the problems of feature extraction and selection. Hough transform, FFT, and textural features, just to mention some, are all examples of features widely applied in image classification and scene understanding tasks. Their properties have been well investigated and available tools make their use simple and efficient.

- *How do we represent visual information?*

  In many computer vision applications, feature vectors are used to represent the perceived environment. However, relational descriptions are deemed to be of crucial importance in high-level vision. Since relations cannot be represented by feature vectors, pattern recognition researchers use graphs to capture the structure of both objects and scenes, while people working in the field of machine learning prefer to use first-order logic formalisms. By mapping one formalism into another, it is possible to find some similarities between research done in pattern recognition and machine learning. An example is the spatio-temporal decision tree proposed by Bischof and Caelli [Bischof and Caelli, 2001], which can be related to logical decision trees induced by some general-purpose inductive learning systems [Blockeel and De Raedt, 1998].

- *What machine learning paradigms and strategies are appropriate to the computer vision domain?*

  Inductive learning, both supervised and unsupervised, emerges as the most important learning strategy. There are several important paradigms that are being used: conceptual (decision trees, graph-induction), statistical (support vector machines), and neural networks (Kohonen maps and similar auto-organizing systems). Another emerging paradigm, which is described in detail in this book, is the use of probabilistic models in general and probabilistic graphical models in particular.

- *What are the criteria for evaluating the quality of the learning processes in computer vision systems?*

  In benchmarking computer vision systems, estimates of the predictive accuracy, recall, and precision [Huijsman and Sebe, 2004] are considered the main parameters to evaluate the success of a learning algorithm. However, the comprehensibility of learned models is also deemed an important criterion, especially when domain experts have strong expectations on the properties of visual models or when understanding of system failures is important. Comprehensibility is needed by the expert to easily and reliably verify the inductive assertions and relate them to their own domain knowledge. When comprehensibility is an important issue, the conceptual learning paradigm is usually preferred, since it is based on the comprehensibility postulate stated by Michalski [Michalski, 1983]:

> The results of computer induction should be symbolic descriptions of given entities, semantically and structurally similar to those a human expert might produce observing the same entities. Components of these descriptions should be comprehensible as single "chunks" of information, directly interpretable in natural language, and should relate quantitative and qualitative concepts in an integrated fashion.

- *When is it useful to adopt several representations of the perceived environment with different levels of abstraction?*

In complex real-world applications, multi-representations of the perceived environment prove very useful. For instance, a low resolution document image is suitable for the efficient separation of text from graphics, while a finer resolution is required for the subsequent step of interpreting the symbols in a text block (OCR). Analogously, the representation of an aerial view of a cultivated area by means of a vector of textural features can be appropriate to recognize the type of vegetation, but it is too coarse for the recognition of a particular geomorphology. By applying abstraction principles in computer programming, software engineers have managed to develop complex software systems. Similarly, the systematic application of abstraction principles in knowledge representation is the keystone for a long term solution to many problems encountered in computer vision tasks.

- *How can mutual dependency of visual concepts be dealt with?*

In scene labelling problems, image segments have to be associated with a class name or a label, the number of distinct labels depending on the different types of objects allowed in the perceived world. Typically, image segments cannot be labelled independently of each other, since the interpretation of a part of a scene depends on the understanding of the whole scene (holistic view). Context-dependent labelling rules will take such concept dependencies into account, so as to guarantee that the final result is globally (and not only locally) consistent [Haralick and Shapiro, 1979]. Learning context-dependent labelling rules is another research issue, since most learning algorithms rely on the independence assumption, according to which the solution to a multiclass or multiple concept learning problem is simply the sum of independent solutions to single class or single concept learning problems.

Obviously, the above list cannot be considered complete. Other equally relevant research issues might be proposed, such as the development of noise-tolerant learning techniques, the effective use of large sets of unlabeled images and the identification of suitable criteria for starting/stopping the learning process and/or revising acquired visual models.

2. Overview of the Book

In general, the study of machine learning and computer vision can be divided into three broad categories: *Theory* leading to *Algorithms* and *Applications* built on top of theory and algorithms. In this framework, the applications should form the basis of the theoretical research leading to interesting algorithms. As a consequence, the book was divided into three parts. The first part develops the theoretical understanding of the concepts that are being used in developing algorithms in the second part. The third part focuses on the analysis of computer vision and human-computer interaction applications that use the algorithms and the theory presented in the first parts.

The theoretical results in this book originate from different practical problems encountered when using machine learning in general, and probabilistic models in particular, to computer vision and multimedia problems. The first set of questions arise from the high dimensionality of models in computer vision and multimedia. For example, integration of audio and visual information plays a critical role in multimedia analysis. Different media streams (e.g., audio, video, and text, etc.) may carry information about the task being performed and recent results [Brand et al., 1997; Chen and Rao, 1998; Garg et al., 2000b] have shown that improved performance can be obtained by combining information from different sources compared with the situation when a single modality is considered. At times, different streams may carry similar information and in that case, one attempts to use the redundancy to improve the performance of the desired task by cancelling the noise. At other times, two streams may carry complimentary information and in that case the system must make use of the information carried in both channels to carry out the task. However, the merits of using multiple streams is overshadowed by the formidable task of learning in high dimensional which is invariably the case in multi-modal information processing. Although, the existing theory supports the task of learning in high dimensional spaces, the data and model complexity requirements posed are typically not met by the real life systems. Under such scenario, the existing results in learning theory falls short of giving any meaningful guarantees for the learned classifiers. This raises a number of interesting questions:

- Can we analyze the learning theory for more practical scenarios?
- Can the results of such analysis be used to develop better algorithms?

Another set of questions arise from the practical problem of data availability in computer vision, mainly labeled data. In this respect, there are three main paradigms for learning from training data. The first is known as *supervised learning*, in which all the training data are labeled, i.e., a datum contains both the values of the attributes and the labeling of the attributes to one of the classes. The labeling of the training data is usually done by an external mechanism (usually humans) and thus the name *supervised*. The second is known as *unsupervised learning* in which each datum contains the values of the attributes but does not contain the label. Unsupervised learning tries to find regularities in the unlabeled training data (such as different clusters under some metric space), infer the class labels and sometimes even the number of classes. The third kind is *semi-supervised learning* in which some of the data is labeled and some unlabeled. In this book, we are more interested in the latter.

Semi-supervised learning is motivated from the fact that in many computer vision (and other real world) problems, obtaining unlabeled data is relatively easy (e.g., collecting images of faces and non-faces), while labeling is difficult, expensive, and/or labor intensive. Thus, in many problems, it is very desirable to have learning algorithms that are able to incorporate a large number of unlabeled data with a small number of labeled data when learning classifiers.

Some of the questions raised in semi-supervised learning of classifiers are:

- Is it feasible to use unlabeled data in the learning process?
- Is the classification performance of the learned classifier guaranteed to improve when adding the unlabeled data to the labeled data?
- What is the value of unlabeled data?

The goal of the book is to address all the challenging questions posed so far. We believe that a detailed analysis of the way machine learning theory can be applied through algorithms to real-world applications is very important and extremely relevant to the scientific community.

Chapters 2, 3, and 4 provide the theoretical answers to the questions posed above. Chapter 2 introduces the basics of probabilistic classifiers. We argue that there are two main factors contributing to the error of a classifier. Because of the inherent nature of the data, there is an upper limit on the performance of any classifier and this is typically referred to as Bayes optimal error. We start by analyzing the relationship between the Bayes optimal performance of a classifier and the conditional entropy of the data. The mismatch between the true underlying model (one that generated the data) and the model used for classification contributes to the second factor of error. In this chapter, we develop bounds on the classification error under the hypothesis testing framework when there is a mismatch in the distribution used with respect to the true distribution. Our bounds show that the classification error is closely related to the conditional entropy of the distribution. The additional penalty, because of the mismatched distribution, is a function of the Kullback-Leibler distance between the true and the mismatched distribution. Once these bounds are developed, the next logical step is to see how often the error caused by the mismatch between distributions is large. Our average case analysis for the independence assumptions leads to results that justify the success of the conditional independence assumption (e.g., in naive Bayes architecture). We show that in most cases, almost all distributions are very close to the distribution assuming conditional independence. More formally, we show that the number of distributions for which the additional penalty term is large goes down exponentially fast.

Roth [Roth, 1998] has shown that the probabilistic classifiers can be always mapped to linear classifiers and as such, one can analyze the performance of these under the probably approximately correct (PAC) or Vapnik-Chervonenkis (VC)-dimension framework. This viewpoint is important as it allows one to directly study the classification performance by developing the relations between the performance on the training data and the expected performance on the future unseen data. In Chapter 3, we build on these results of Roth [Roth, 1998]. It turns out that although the existing theory argues that one needs large amounts of data to do the learning, we observe that in practice a good generalization is achieved with a much small number of examples. The existing VC-dimension based bounds (being the worst case bounds) are too loose and we need to make use of properties of the observed data leading to data dependent bounds. Our observation, that in practice, classification is achieved with good margin, motivates us to develop bounds based on margin distribution. We develop a classification version of the Random projection theorem [Johnson and Lindenstrauss, 1984] and use it to develop data dependent bounds. Our results show that in most problems of practical interest, data actually reside in a low dimensional space. Comparison with existing bounds on real datasets shows that our bounds are tighter than existing bounds and in most cases less than 0.5.

The next chapter (Chapter 4) provides a unified framework of probabilistic classifiers learned using maximum likelihood estimation. In a nutshell, we discuss what type of probabilistic classifiers are suited for using unlabeled data in a systematic way with the maximum likelihood learning, namely classifiers known as *generative*. We discuss the conditions under which the assertion that unlabeled data are always profitable when learning classifiers, made in the existing literature, is valid, namely when the assumed probabilistic model matches reality. We also show, both analytically and experimentally, that unlabeled data can be detrimental to the classification performance when the conditions are violated. Here we use the term ‘reality’ to mean that there exists some true probability distribution that generates data, the same one for both labeled and unlabeled data. The terms are more rigourously defined in Chapter 4.

The theoretical analysis although interesting in itself gets really attractive if it can be put to use in practical problems. Chapters 5 and 6 build on the results developed in Chapters 2 and 3, respectively. In Chapter 5, we use the results of Chapter 2 to develop a new algorithm for learning HMMs. In Chapter 2, we show that conditional entropy is inversely related to classification performance. Building on this idea, we argue that when HMMs are used for classification, instead of learning parameters by only maximizing the likelihood, one should also attempt to minimize the conditional entropy between the query (hidden) and the observed variables. This leads to a new algorithm for learning HMMs - MMIHMM. Our results on both synthetic and real data demonstrate the superiority of this new algorithm over the standard ML learning of HMMs.

In Chapter 3, a new, data-dependent, complexity measure for learning – projection profile – is introduced and is used to develop improved generalization bounds. In Chapter 6, we extend this result by developing a new learning algorithm for linear classifiers. The complexity measure – *projection profile* – is a function of the *margin distribution* (the distribution of the distance of instances from a separating hyperplane). We argue that instead of maximizing the margin, one should attempt to directly minimize this term which actually depends on the margin distribution. Experimental results on some real world problems (face detection and context sensitive spelling correction) and on several UCI data sets show that this new algorithm is superior (in terms of classification performance) over Boosting and SVM.

Chapter 7 provides a discussion of the implication of the analysis of semi-supervised learning (Chapter 4) when learning Bayesian network classifiers, suggesting and comparing different approaches that can be taken to utilize positively unlabeled data. Bayesian networks are directed acyclic graph models that represent joint probability distributions of a set of variables. The graphs consist of nodes (vertices in the graph) which represent the random variables and directed edges between the nodes which represent probabilistic dependencies between the variables and the casual relationship between the two connected nodes. With each node there is an associated probability mass function when the variable is discrete, or probability distribution function, when the variable is continuous. In classification, one of the nodes in the graph is the class variable while the rest are the attributes. One of the main advantages of Bayesian networks is the ability to handle missing data, thus it is possible to systematically handle unlabeled data when learning the Bayesian network. The structure of a Bayesian network is the graph structure of the network. We show that learning the graph structure of the Bayesian network is key when learning with unlabeled data. Motivated by this observation, we review the existing structure learning approaches and point out to their potential disadvantages when learning classifiers. We describe a structure learning algorithm, driven by classification accuracy and provide empirical evidence of the algorithm's success.

Chapter 8 deals with automatic recognition of high level human behavior. In particular, we focus on the office scenario and attempt to build a system that can decode the human activities (*phone conversation, face-to-face conversation, presentation mode, other activity, nobody around,* and *distant conversation*). Although there has been some work in the area of behavioral analysis, this is probably the first system that does the automatic recognition of human activities in real time from low-level sensory inputs. We make use of probabilistic models for this task. Hidden Markov models (HMMs) have been successfully applied for the task of analyzing temporal data (e.g. speech). Although very powerful, HMMs are not very successful in capturing the long term relationships and modeling concepts lasting over long periods of time. One can always increase the number of hidden states but then the complexity of decoding and the amount of data required to learn increases many fold. In our work, to solve this problem, we propose the use of layered (a type of hierarchical) HMMs (LHMM), which can be viewed as a special case of Stacked Generalization [Wolpert, 1992]. At each level of the hierarchy, HMMs are used as classifiers to do the inference. The inferential output of these HMMs forms the input to the next level of the hierarchy. As our results show, this new architecture has a number of advantages over the standard HMMs. It allows one to capture events at different level of abstraction and at the same time is capturing long term dependencies which are critical in the modeling of higher level concepts (human activities). Furthermore, this architecture provides robustness to noise and generalizes well to different settings. Comparison with standard HMM shows that this model has superior performance in modeling the behavioral concepts.

The other challenging problem related to multimedia deals with automatic analysis/annotation of videos. This problem forms the topic of Chapter 9. Although similar in spirit to the problem of human activity recognition, this problem gets challenging because of the limited number of modalities (audio and vision) and the correlation between them being the key in event identification. In this chapter, we present a new algorithm for detecting events in videos, which combines the features with temporal support from multiple modalities. This algorithm is based on a new framework "Duration dependent input/output Markov models (DDIOMM)". Essentially DDIOMM is a time varying Markov model (state transition matrix is a function of the inputs at any given time) and the state transition probabilities are modified to explicitly take into account the non-exponential nature of the durations of various events being modeled. Two main features of this model are (a) the ability to account for non-exponential duration and (b) the ability to map discrete state input sequences to decision sequences. The standard algorithms modeling the video-events use HMMs which model the duration of events as an exponentially decaying distribution. However, we argue that the duration is an important characteristic of each event and we demonstrate it by the improved performance over standard HMMs in solving real world problems. The model is tested on the audio-visual event *explosion*. Using a set of hand-labeled video data, we compare the performance of our model with and without the explicit model for duration. We also compare the performance of the proposed model with the traditional HMM and observe an improvement in detection performance.

The algorithms LHMM and DDIOMM presented in Chapters 8 and 9, respectively, have their origins in HMM and are motivated by the vast literature on probabilistic models and some psychological studies arguing that human behavior does have a hierarchical structure [Zacks and Tversky, 2001]. However, the problem lies in the fact that we are using these probabilistic models for classification and not purely for inferencing (the performance is measured with respect to the $0-1$ loss function). Although one can use arguments related to Bayes optimality, these arguments fall apart in the case of mismatched distributions (i.e. when the true distribution is different from the used one). This mismatch may arise because of the small number of training samples used for learning, assumptions made to simplify the inference procedure (e.g. a number of conditional independence assumptions are made in Bayesian networks) or may be just because of the lack of information about the true model. Following the arguments of Roth [Roth, 1999], one can analyze these algorithms both from the perspective of probabilistic classifiers and from the perspective of statistical learning theory. We apply these algorithms to two distinct but related applications which require machine learning techniques for multimodal information fusion: office activity recognition and multimodal event detection.

Chapters 10 and 11 demonstrate the theory and algorithms of semi-supervised learning (Chapters 4 and 7) to two classification tasks related to human computer intelligent interaction. The first is facial expression recognition from video sequences using non-rigid face tracking results as the attributes. We show that Bayesian networks can be used as classifiers to recognize facial expressions with good accuracy when the structure of the network is estimated from data. We also describe a real-time facial expression recognition system which is based on this analysis. The second application is frontal face detection from images under various illuminations. We describe the task and show that learning Bayesian network classifiers for detecting faces using our structure learning algorithm yields improved classification results, both in the supervised setting and in the semi-supervised setting.

3. Contributions

Original contributions presented in this book span the areas of learning architectures for multimodal human computer interaction, theoretical machine learning, and algorithms in the area of machine learning. In particular, some key issues addressed in this book are:

Theory

- Analysis of probabilistic classifiers leading to developing relationship between the Bayes optimal error and the conditional entropy of the distribution.
- Bounds on the misclassification error under $0-1$ loss function are developed for probabilistic classifiers under hypothesis testing framework when there is a mismatch between the true distribution and the learned distribution.
- Average case analysis of the space of probability distributions. Results obtained show that almost all distributions in the space of probability distributions are close to the distribution that assumes conditional independence between the features given the class label.
- Data dependent bounds are developed for linear classifiers that depend on the margin distribution of the data with respect to the learned classifier.
- An extensive discussion of using labeled and unlabeled data for learning probabilistic classifiers. We discuss the types of probabilistic classifiers that are suited for using unlabeled data in learning and we investigate the conditions under which the assertion that unlabeled data are always profitable when learning classifiers is valid.

Algorithms

- A new learning algorithm MMIHMM (Maximum mutual information HMM) for hidden Markov models is proposed when HMMs are used for classification with states as hidden variables.
- A novel learning algorithm - Margin Distribution optimization algorithm is introduced for learning linear classifiers.
- New algorithms for learning the structure of Bayesian Networks to be used in semi-supervised learning.

Applications

- A novel architecture for human activity recognition - Layered HMM - is proposed. This architecture allows one to model activities by combining heterogeneous sources and analyzing activities at different levels of temporal abstraction. Empirically, this architecture is observed to be robust to environmental noise and provides good generalization capabilities in different settings.

- A new architecture based on HMMs is proposed for detecting events in videos. Multimodal events are characterized by the correlation in different media streams and their specific durations. This is captured by the new architecture Duration density Hidden Markov Model proposed in the book.

- A Bayesian Networks framework for recognizing facial expressions from video sequences using labeled and unlabeled data is introduced. We also present a real-time facial expression recognition system.

- An architecture for frontal face detection from images under various illuminations is presented. We show that learning Bayesian Networks classifiers for detecting faces using our structure learning algorithm yields improved classification results both in the supervised setting and in the semi-supervised setting.

This book concentrates on the application domains of human-computer interaction, multimedia analysis, and computer vision. However the results and algorithms presented in the book are general and equally applicable to other areas including *speech recognition, content-based retrieval, bioinformatics,* and *text processing*. Finally, the chapters in this book are mostly self contained; each chapter includes self consistent definitions and notations meant to ease the reading of each chapter in isolation.

Chapter 2

THEORY: PROBABILISTIC CLASSIFIERS

Probabilistic classifiers are developed by assuming generative models which are product distributions over the original attribute space (as in naive Bayes) or more involved spaces (as in general Bayesian networks). While this paradigm has been shown experimentally successful on real world applications, despite vastly simplified probabilistic assumptions, the question of why these approaches work is still open.

The goal of this chapter is to give an answer to this question. We show that almost all joint distributions with a given set of marginals (i.e., all distributions that could have given rise to the classifier learned) or, equivalently, almost all data sets that yield this set of marginals, are very close (in terms of distributional distance) to the product distribution on the marginals; the number of these distributions goes down exponentially with their distance from the product distribution. Consequently, as we show, for almost all joint distributions with this set of marginals, the penalty incurred in using the marginal distribution rather than the true one is small. In addition to resolving the puzzle surrounding the success of probabilistic classifiers, our results contribute to understanding the tradeoffs in developing probabilistic classifiers and help in developing better classifiers.

1. Introduction

Probabilistic classifiers and, in particular, the archetypical naive Bayes classifier, are among the most popular classifiers used in the machine learning community and increasingly in many applications. These classifiers are derived from generative probability models which provide a principled way to the study of statistical classification in complex domains such as natural language and visual processing.

The study of probabilistic classification is the study of approximating a joint distribution with a product distribution. Bayes rule is used to estimate the conditional probability of a class label y, and then assumptions are made on the model, to decompose this probability into a product of conditional probabilities:

$$\begin{aligned} P(y|x) = P(y|x^1, x^2, \ldots x^n) &= \prod_{i=1}^{n} P(x^i|x^1, \ldots x^{i-1}, y)\frac{P(y)}{P(x)} \\ &= \prod_{j=1}^{n'} P(y^j|y)\frac{P(y)}{P(x)}, \end{aligned} \tag{2.1}$$

where $x = (x^1, \ldots, x^n)$ is the observation and the $y^j = g_j(x^1, \ldots x^{i-1}, x^i)$, for some function g_j, are independent given the class label y.

While the use of Bayes rule is harmless, the final decomposition step introduces independence assumptions which may not hold in the data. The functions g_j encode the probabilistic assumptions and allow the representation of any Bayesian network, e.g., a Markov model. The most common model used in classification, however, is the *naive Bayes* model in which $\forall j, g_j(x^1, \ldots x^{i-1}, x^i) \equiv x^i$. That is, the original attributes are assumed to be independent given the class label.

Although the naive Bayes algorithm makes some unrealistic probabilistic assumptions, it has been found to work remarkably well in practice [Elkan, 1997; Domingos and Pazzani, 1997]. Roth [Roth, 1999] gave a partial answer to this unexpected behavior using techniques from learning theory. It is shown that naive Bayes and other probabilistic classifiers are all “Linear Statistical Query” classifiers; thus, PAC type guarantees [Valiant, 1984] can be given on the performance of the classifier on future, previously unseen data, as a function of its performance on the training data, independently of the probabilistic assumptions made when deriving the classifier. However, the key question that underlies the success of probabilistic classifiers is still open. That is, why is it even possible to get good performance on the training data, i.e., to “fit the data”[1] with a classifier that relies heavily on extremely simplified probabilistic assumptions on the data?

This chapter resolves this question and develops arguments that could explain the success of probabilistic classifiers and, in particular, that of naive Bayes. The results are developed by doing the combinatoric analysis on the space of all distributions satisfying some properties.

[1] We assume here a fixed feature space; clearly, by blowing up the feature space it is always possible to fit the data.

One important point to note is that in this analysis we have made use of the counting arguments to derive most of the results. What that means is that we look at the space of all distributions, where distributions are quantized in some sense (which will be made clear in the respective context), and then we look at these finite number of points (each distribution can be thought of as a point in the distribution space), and try to quantify the properties of this space. This is very different from assuming the uniform prior distribution over the distribution space as this allows our results to be extended to any prior distribution.

This chapter starts by quantifying the optimal Bayes error as a function of the entropy of the data conditioned upon the class label. We develop upper and lower bounds on this term (give the feasible region), and discuss where do most of the distributions lie relative to these bounds. While this gives some idea as to what can be expected in the best case, one would like to quantify what happens in realistic situations, when the probability distribution is not known. Normally in such circumstances one ends up making a number of independence assumption. Quantifying the penalty incurred due to the independence assumptions allows us to show its direct relation to the distributional distance between the true (joint) and the product distribution over the marginals used to derive the classifier. This is used to derive the main result of the chapter which, we believe, explains the practical success of product distribution based classifiers. Informally, we show that *almost all* joint distributions with a given set of marginals (that is, all distributions that could have given rise to the classifier learned)[2] are very close to the product distribution on the marginals - the number of these distributions goes down exponentially with their distance from the product distribution. Consequently, the error incurred when predicting using the product distribution is small for *almost all* joint distributions with the same marginals.

There is no claim in this chapter that distributions governing "practical" problems are sampled according to a uniform distribution over these marginal distributions. Clearly, there are many distributions for which the product distribution based algorithm will not perform well (e.g., see [Roth, 1999]) and in some situations, these could be the interesting distributions. The counting arguments developed here suggest, though, that "bad" distributions are relatively rare.

Finally, we show how these insights may allow one to quantify the potential gain achieved by the use of complex probabilistic models thus explaining phenomena observed previously by experimenters.

[2] Or, equivalently, as we show, almost all data sets with this set of marginals.

It is important to note that this analysis ignores small sample effects. We do not attend to learnability issues but rather assume that good estimates of the statistics required by the classifier can be obtained; the chapter concentrates on analyzing the properties of the resulting classifiers.

2. Preliminaries and Notations

Throughout this chapter we will use capital letter to denote random variables and the same token in lower case (x, y, z) to denote particular instantiations of them. $P(x|y)$ will denote the probability of random variable X taking on value x, given that the random variable Y takes the value y. X^i denotes the i^{th} component of the random vector X. For a probability distribution P, $P^{[n]}(\cdot)$ denotes the joint probability of observing a sequence of n i.i.d samples distributed according to P.

Throughout the chapter we consider random variables over a discrete domain $\mathcal{X}$, of size $|\mathcal{X}| = N$, or over $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{Y}$ is also discrete and typically, $|\mathcal{Y}| = 2$. In these cases, we typically denote by $\mathcal{X} = \{0, 1, ..., N-1\}, \mathcal{Y} = \{0, 1\}$.

Definition 2.1 *Let $X = (X^1, X^2, ..., X^n)$ be a random vector over $\mathcal{X}$, distributed according to Q. The* marginal distribution *of the ith component of X, denoted Q^i, is a distribution over X^i, given by*

$$Q^i(x) = \sum_{x^j \in X^j; \forall j \neq i} Q(x^1, \ldots x^{i-1}, x^i, x^{i+1}, \ldots x^n). \tag{2.2}$$

The product distribution *induced by Q over $\mathcal{X}$ is given by*

$$Q_m(x) = \prod_{i=1}^{n} Q^i(x^i). \tag{2.3}$$

Note that Q_m is identical to Q when assuming that in Q, the components X^i of X are independent of each other. We sometimes call Q_m the marginal distribution *induced by Q.*

2.1 Maximum Likelihood Classification

We consider the standard binary classification problem in a probabilistic setting. This model assumes that data elements (x, y) are sampled according to some arbitrary distribution P on $\mathcal{X} \times \{0, 1\}$. $\mathcal{X}$ is the *instance space* and $y \in \{0, 1\}$ is called the *class label*. The goal of the learner is to determine, given a new example $x \in \mathcal{X}$, its most likely corresponding label $y(x)$, which is chosen as follows:

$$y(x) = \underset{i \in \{0,1\}}{argmax}\, P(y = i|x) = \underset{i \in \{0,1\}}{argmax}\, P(x|y = i) \frac{P(y = i)}{P(x)}. \tag{2.4}$$

Given the distribution P on $\mathcal{X} \times \{0,1\}$, we define the following distributions over $\mathcal{X}$:

$$P_0 \doteq P(x|y=0) \quad \text{and} \quad P_1 \doteq P(x|y=1). \tag{2.5}$$

With this notation, the Bayesian classifier (in Eqn 2.4) predicts $y = 1$ if and only if $P_0(x) < P_1(x)$.

When $\mathcal{X} = \{0,1\}^n$ (or any other discrete product space) we will write $x = (x^1, \ldots x^n) \in \mathcal{X}$, and denote a sample of elements in $\mathcal{X}$ by $S = \{x_1, \ldots x_m\} \subseteq \mathcal{X}$, with $|S| = m$. The sample is used to estimate $P(x|y)$, which is approximated using a conditional independence assumption:

$$P(x|y) = P(x^1, \ldots x^n|y) = \prod_{i=1}^{n} P(x^i|y). \tag{2.6}$$

Using the conditional independence assumption, the prediction in Eqn 2.4 is done by estimating the product distributions induced by P_0 and P_1,

$$P_{m0} = \prod_{i=1}^{n} P(x^i|y=0) \quad \text{and} \quad P_{m1} = \prod_{i=1}^{n} P(x^i|y=1), \tag{2.7}$$

and predicting $y(x) = 1$ iff

$$p(y=0)P_{m0}(x) \leq p(y=1)P_{m1}(x). \tag{2.8}$$

This is typically referred to as the naive Bayes methods of classification [Duda and Hart, 1973].

2.2 Information Theory

Definition 2.2 (Entropy; Kullback-Leibler Distance) *Let X be a random variable over $\mathcal{X}$, distributed according to P.*

The entropy of X (sometimes written as "the entropy of P") is given by

$$H(X) = H(P) = -\sum_{x \in \mathcal{X}} P(x) \log P(x) \tag{2.9}$$

where the log is to the base 2. Note that the entropy of X can also be interpreted as the expected value of $\log \frac{1}{P(X)}$, which is a function of random variable X drawn according to P.

The joint entropy $H(X,Y)$ of a pair of discrete random variables (X,Y) with a joint distribution $P(x,y)$ is defined as

$$H(X,Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P(x,y) \log P(x,y). \tag{2.10}$$

and the conditional entropy $H(X|Y)$ of X given Y is defined as

$$H(X|Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(x,y)\log P(x|y). \tag{2.11}$$

Let P, Q be two probability distributions over a discrete domain $\mathcal{X}$. The *relative entropy* or the *Kullback-Leibler distance* between P and Q is defined as

$$D(P||Q) = \sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)}{Q(x)} = E_p \log\frac{P(X)}{Q(X)}. \tag{2.12}$$

2.3 Inequalities

1 (**Jensen's Inequality**)([Cover and Thomas, 1991], p. 25) If f is a convex function and X is a random variable, then

$$E[f(X)] \geq f(E[X]). \tag{2.13}$$

2 For any probability density function P over domain $\mathcal{X} = \{1, 2, ..., N\}$ we have

$$H(P) = E_P(-\log P(X)) = -\sum_{i=1}^{N} p_i \log p_i \geq -\log\sum_{i=1}^{N} p_i^2 \tag{2.14}$$

which follows from Jensen's inequality using the convexity of $-\log(x)$, applied to the random variable $p(x)$, where $X \sim p(x)$.

3 For any $x, k > 0$, we have

$$1 + \log k - kx \leq -\log x \tag{2.15}$$

which follows from $\log(x) \leq x - 1$ by replacing x by kx. Equality holds when $k = 1/x$. Equivalently, replacing x by e^{-x} we have

$$1 - x \leq e^{-x}. \tag{2.16}$$

For more details please see [Cover and Thomas, 1991].

3. Bayes Optimal Error and Entropy

In this section, we are interested in the optimal error achievable by a Bayes classifier (Eqn 2.4) on a sample $\{(x, y)\}_1^m$ sampled according to a distribution P over $\mathcal{X} \times \{0, 1\}$. At this point no independence assumption is made and the results in this section apply to any Maximum likelihood classifier as defined in Eqn 2.4. For simplicity of analysis, we restrict our discussion to the equal class probability case, $P(y = 1) = P(y = 0) = \frac{1}{2}$. The optimal Bayes error is defined by

$$\epsilon = \frac{1}{2}P_0(\{x|P_1(x) > P_0(x)\}) + \frac{1}{2}P_1(\{x|P_0(x) > P_1(x)\}), \tag{2.17}$$

and the following result relates it to the distance between P_0 and P_1:

LEMMA 2.3 *([Devroye et al., 1996], p. 15) The Bayes optimal error under the equal class probability assumption is:*

$$\epsilon = \frac{1}{2} - \frac{1}{4}\sum_x |P_0(x) - P_1(x)|. \tag{2.18}$$

Note that $P_0(x)$ and $P_1(x)$ are "independent" quantities. Theorem 3.2 from [Devroye et al., 1996] also gives the relation between the Bayes optimal error and the entropy of the class label (random variable $Y \in \mathcal{Y}$) conditioned upon the data $X \in \mathcal{X}$:

$$-\log(1-\epsilon) \leq H(Y|X) \equiv H(P(y|x)) \leq -\epsilon\log\epsilon - (1-\epsilon)\log(1-\epsilon). \tag{2.19}$$

However, the availability of $P(y|x)$ typically depends on first learning a probabilistic classifier which might require a number of assumptions. In what follows, we develop results that relate the lowest achievable Bayes error and the conditional entropy of the *input data* given the class label, thus allowing an assessment of the optimal performance of the Bayes classifier directly from the given data. Naturally, this relation is much looser than the one given in Eqn 2.19, as has been documented in previous attempts to develop bounds of this sort [Feder and Merhav, 1994]. Let $H_b(p)$ denote the entropy of the distribution $\{p, 1-p\}$:

$$H_b(p) = -(1-p)\log(1-p) - p\log p.$$

THEOREM 2.4 *Let $X \in \mathcal{X}$ denote the feature vector and $Y \in \mathcal{Y}$, denote the class label, then under equal class probability assumption, and an optimal Bayes error of ϵ, the conditional entropy $H(X|Y)$ of input data conditioned upon the class label is bounded by*

$$\frac{1}{2}H_b(2\epsilon) \leq H(X|Y) \leq H_b(\epsilon) + \log\frac{N}{2}. \tag{2.20}$$

We prove the theorem using the following sequence of lemmas. For simplicity, our analysis assumes that N is an even number. The general case follows similarly. In the following lemmas we consider two probability distributions P, Q defined over $\mathcal{X} = \{0, 1, \ldots, N-1\}$. Let $p_i = P(x = i)$ and $q_i = Q(x = i)$. Without losing generality, we assume that $\forall i, j : 0 \leq i, j < N$, when $i < j$, $p_i - q_i > p_j - q_j$ (which can always be achieved by renaming the elements of $\mathcal{X}$).

LEMMA 2.5 *Consider two probability distributions* P, Q *defined over* $\mathcal{X} = \{0, 1, \ldots, N-1\}$. *Let* $p_i = P(x = i)$ *and* $q_i = Q(x = i)$ *and denote* $\sum_i |p_i - q_i| = \alpha$. *Then, the sum* $H(P) + H(Q)$ *obtains its maximal value when for some constants* c_1, c_2, d_1, d_2 *(which depend on* α, K, N*),* P *and* Q *satisfy:*

$$\begin{aligned} &\forall i, 0 \le i \le M/2 && p_i = c_1, q_i = d_1 \ \textit{and} \\ &\forall i, M/2 < i \le N && p_i = c_2, \ \ q_i = d_2. \end{aligned} \tag{2.21}$$

Proof. We will first show that, $H(P) + H(Q)$ obtains its maximum value for some K, such that

$$\forall i, 0 \le i \le K \ \ p_i = c_1, q_i = d_1 \ \text{ and } \ \forall i, K < i \le N \ \ p_i = c_2, \ \ q_i = d_2,$$

and will then show that this maximum is achieved for $K = M/2$.

We want to maximize the function

$$-\sum_i p_i \log p_i - \sum_i q_i \log q_i$$

subject to the constraints

$$\sum_i |p_i - q_i| = \alpha, \quad \sum_i p_i = 1, \quad \sum_i q_i = 1.$$

The Lagrange formulation for the above optimization problem can be written as

$$\begin{aligned} \lambda = \quad & -\sum_i p_i \log p_i - \sum_i q_i \log q_i \\ & + a\left(\sum_i |p_i - q_i| - \alpha\right) + b\left(\sum_i p_i - 1\right) + c\left(\sum_i q_i - 1\right), \end{aligned}$$

where a, b, c are Lagrange multipliers. When differentiating λ with respect to p_i and q_i, we obtain that the sum of the two entropies is maximized when, for some constants A, B, C,

$$\forall i : 0 \le i \le K, p_i = A\exp(C), \ \ q_i = B\exp(-C),$$

and

$$\forall i : K < i \le N, p_i = A\exp(-C), \ \ q_i = B\exp(C)$$

where $0 \le K \le N$ is the largest index such that $p_i - q_i > 0$.

Denote $p = \sum_{i=1}^{K} p_i$ and $q = \sum_{i=1}^{K} q_i$. Then, we have $(p-q) + ((1-q) - (1-p)) = \alpha$, giving

$$\begin{array}{lll} 0 \leq i \leq K, & p_i = \frac{p}{K}, & q_i = p_i - \frac{\alpha}{2K} \\ K < i \leq N, & p_i = \frac{1-p}{N-K}, & q_i = p_i + \frac{\alpha}{2(N-K)}. \end{array}$$

These distributions maximize $H(P) + H(Q)$, which can now be written as

$$\begin{aligned} H(P){+}H(Q) = -p \log \frac{p}{K} - (1-p) \log \frac{1-p}{N-K} \\ -(p-\alpha/2) \log \frac{p-\alpha/2}{K} - (1-p+\alpha/2) \log \frac{1-p+\alpha/2}{N-K}. \end{aligned}$$

Differentiating the above expression with respect to K and with respect to p, the maximum is achieved when $K = N/2$ and

$$p = \frac{1+\alpha/2}{2}, \quad q = \frac{1-\alpha/2}{2}. \tag{2.22}$$

The next lemma is used later to develop the lower bound on the conditional entropy.

LEMMA 2.6 *Consider two probability distributions P, Q defined over $\mathcal{X} = \{0, 1, \ldots, N-1\}$. Let $p_i = P(x = i)$ and $q_i = Q(x = i)$ and denote $\sum_i |p_i - q_i| = \alpha$. The sum $H(P) + H(Q)$ of their entropies is minimized when all the mass of Q is on single instance i (i.e. $q_i = 1$) with for same i, $p_i = 1 - \alpha/2$ and for some $j \neq i$, $p_j = \alpha/2$. That is,*

$$\begin{aligned} P &= \{0, .., 0, p_j = \frac{\alpha}{2}, 0, ..., 0, p_i = 1 - \frac{\alpha}{2}, 0, .., 0\} \text{ and} \\ Q &= \{0, .., 0, 0, 0, ..., 0, p_i = 1, 0, ..., 0\}. \end{aligned}$$

Proof. Note that for any set of non negative numbers $a_1, a_2, ...,$

$$\sum_i a_i \log a_i \leq \sum_i a_i \log \sum_j a_j \leq \left(\sum_i a_i\right) \log \left(\sum_i a_i\right).$$

We want to minimize the quantity:

$$H = -\sum_i p_i \log p_i - \sum_i q_i \log q_i,$$

under the constraint $\sum_i |p_i - q_i| = \alpha$. As before (Lemma 2.5), assume that for $0 \leq i \leq K$, $p_i \geq q_i$ and for $K < i \leq N-1$, $p_i \leq q_i$, where $K \in \{0, ..., N-1\}$. This implies:

$$\begin{aligned}
H &= -\sum_i p_i \log p_i - \sum_i q_i \log q_i \\
&= -\sum_{i=0}^{K} p_i \log p_i - \sum_{i=K+1}^{N-1} p_i \log p_i - \sum_{i=0}^{K} q_i \log q_i - \sum_{i=K+1}^{N-1} q_i \log q_i \\
&\geq -p \log p - (1-p)\log(1-p) - q\log q - (1-q)\log(1-q) \quad (2.23)
\end{aligned}$$

where $p = \sum_{i=0}^{K} p_i$ and $q = \sum_{i=0}^{K} q_i$.

The equality in Eqn 2.22 is achieved if for some $0 \leq j \leq K$, $p_j = p$ and $\forall i : i \neq j, 0 \leq i \leq K,\ p_i = 0$.

In a similar manner, one can write the same equations for $(1-p)$, q, and $(1-q)$. The constraint on the difference of the two distribution, forces that $p - q + (1-q) - (1-p) = \alpha$ which implies that $p = q + \frac{\alpha}{2}$. Under this, we can write H as:

$$\begin{aligned}
H = &- (q+\alpha/2)\log(q+\alpha/2) - (1-q-\alpha/2)\log(1-q-\alpha/2) \\
&- q\log q - (1-q)\log(1-q).
\end{aligned}$$

In the above expression, H is a concave function of q and the minimum (of H) is achieved when either $q = 0$ or $q = 1$. By symmetry $p = \frac{\alpha}{2}$ and $q = 0$.

Now we are in a position to prove Theorem 2.4. Lemma 2.5 is used to prove the upper bound and Lemma 2.6 is used to prove the lower bound on the entropy.

Proof. (**Theorem 2.4**) Assume that $P(y=0) = P(y=1) = \frac{1}{2}$ (equal class probability) and a Bayes optimal error of ϵ. For the upper bound on $H(X|Y)$ we would like to obtain P_0 and P_1 that achieve the maximum conditional entropy. Since

$$\begin{aligned}
H(x|y) &= P(y=0)H(P_0(x)) + P(y=1)H(P_1(x)) \\
&= \frac{1}{2}H(P_0(x)) + \frac{1}{2}H(P_1(x)). \quad (2.24)
\end{aligned}$$

Defining $P_0 \doteq P$ and $P_1 \doteq Q$, the conditional entropy is maximized when the sum of $H(P(x)) + H(Q(x))$ is maximized. The Bayes optimal error of ϵ constrains the two distribution to satisfy $\sum_x |P_0(x) - P_1(x)| = 2 - 4\epsilon = \alpha$. Using the result of Lemma 2.5, we obtain the distributions that maximize the conditional entropy as:

$$\begin{aligned}
P_0 &= \left\{ \frac{1+\frac{\alpha}{2}}{N}, \frac{1+\frac{\alpha}{2}}{N}, ..., \frac{1-\frac{\alpha}{2}}{N}, \frac{1-\frac{\alpha}{2}}{N} \right\} \\
P_1 &= \left\{ \frac{1-\frac{\alpha}{2}}{N}, \frac{1-\frac{\alpha}{2}}{N}, ..., \frac{1+\frac{\alpha}{2}}{N}, \frac{1+\frac{\alpha}{2}}{N} \right\}.
\end{aligned}$$

Note that because of the special form of the above distributions, $H(P) = H(Q)$. The conditional entropy $H(X|Y)$ is given by

$$\begin{aligned} H(x|y) &= \frac{1}{2}H(P_0) + \frac{1}{2}H(P_1) = H(P_0) \\ &= -\frac{1+\alpha/2}{2}\log\frac{1/2+\alpha/4}{N/2} - \frac{1-\alpha/2}{2}\log\frac{1/2-\alpha/4}{N/2} \\ &= -\frac{1+\alpha/2}{2}\log(1/2+\alpha/4) - \frac{1-\alpha/2}{2}\log(1/2-\alpha/4) + \log\frac{N}{2} \\ &= -(1-\epsilon)\log(1-\epsilon) - (\epsilon)\log(\epsilon) + \log\frac{N}{2} \\ &= H_b(\epsilon) + \log\frac{N}{2}. \end{aligned}$$

Lemma 2.6 is used to prove the lower bound on the conditional entropy given Bayes optimal error of ϵ. The choice of distributions in this case is

$$\begin{aligned} P_0 &= \{0, 0, ..., 1 - \frac{\alpha}{2}, ..., 0, \frac{\alpha}{2}, 0, ..., 0\} \text{ and} \\ P_1 &= \{0, 0, ..., 1, ..., 0, 0, 0, ..., 0\}. \end{aligned}$$

The conditional entropy in this case is given by

$$\begin{aligned} H(x|y) &= \frac{1}{2}H(P_0) + \frac{1}{2}H(P_1) \\ &= -(1-\alpha/2)\log(1-\alpha/2) - (\alpha/2)\log(\alpha/2) + 0 \\ &= -(2\epsilon)\log(2\epsilon) - (1-2\epsilon)\log(1-2\epsilon) \\ &= H_b(2\epsilon). \end{aligned}$$

The results of the theorem are depicted in Figure 2.1 for $|\mathcal{X}| = N = 4$. The x-axis gives the conditional entropy of a distribution and the y-axis gives the corresponding range of the Bayes optimal error that can be achieved. The bounds just obtained, imply that the points outside the shaded area, in the figure, cannot be realized. Note that these are tight bounds, in the sense that there are distributions on the boundary of the curves (bounding the shaded region). Interestingly, it also addresses the common misconception that "low entropy implies low error and high entropy implies high error". Our analysis shows that while the latter is correct, the former may not be. That is, it is possible to come up with a distribution with extremely low conditional entropy and still have high Bayes optimal error. However, it does say that if the conditional entropy is high, then one is going to make large error. We observe that when the conditional entropy is zero, the error can either be 0 (no error, perfect classifier, point (A) on graph) or 50% error (point (B) on graph). Although somewhat counterintuitive, consider the following example.

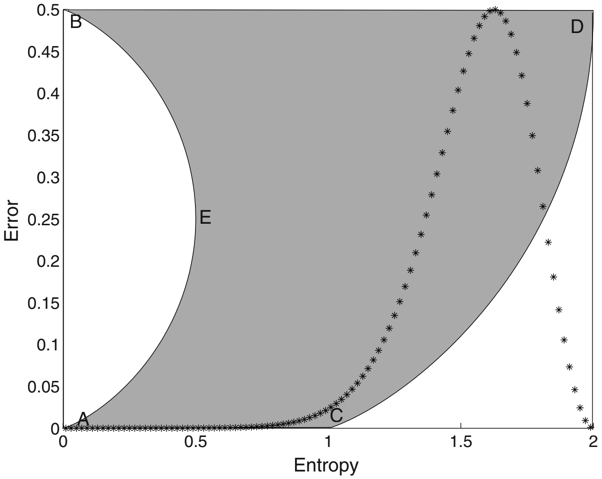


Figure 2.1. The relation between the error and the conditional entropy of the data, for N=4. Here the x-axis gives the conditional entropy and the y-axis gives the range of the Bayes optimal error. The shaded region represents the feasible region (the distributions with the corresponding error and entropy are realizable). The dotted curve gives the empirical distribution of the joint distributions over a given set of input features.

EXAMPLE 2.7 *Let* $P_0(x = 1) = 1$ *and* $P_0(x = i) = 0, \forall i \neq 1$ *and* $\forall x, P_1(x) = P_0(x)$*. Then* $H(x|y) = 0$ *since* $H(P_0(x)) = H(P_1(x)) = 0$ *and the probability of error is* 0.5.

The other critical points on this curve are also realizable. Point (D), which corresponds to the maximum entropy is achieved only when $\forall x,\ \ P_0(x) = \frac{1}{N}$ and $P_1(x) = \frac{1}{N}$. Again the error is 0.5. Point (C) corresponds to the maximum entropy with 0 achievable error. It is given by $H(P(y|x)) = \log\frac{N}{2}$. Finally, point (E) corresponds to the minimum entropy for which there exists a distribution for any value of optimal error. This corresponds to $entropy = 0.5$. Continuity arguments imply that all the shaded area is realizable. At a first glance it appears that the points (A) and (C) are very far apart, as (A) corresponds to 0 entropy whereas (C) corresponds to entropy of $\log\frac{N}{2}$. One might think that most of the joint probability distributions are going to be between (A) and (C) - a range for which the bounds are vacuous. It turns out, however, that most of the distributions actually lie beyond the $\log\frac{N}{2}$ entropy point.

THEOREM 2.8 *Consider a probability distribution over* $x \in \{0, 1, ..., N-1\}$ *given by* $P = [p_0, ..., p_{N-1}]$*, where* $p_i = P(x = i)$*, and assume that* $H(p) \leq \log\frac{N}{2}$*. Then,* $\forall\delta :\ \ 0 < \delta < \frac{1}{N}$*, the distribution Q defined by* $q_i = \frac{1}{N} + \delta(p_i - \frac{1}{N}),\ \ \forall i$ *satisfies* $H(Q) > \log\frac{N}{2}$.

Proof. First note that $\sum_i q_i = 1$ and for $0 < \delta < 1, \forall i,\ q_i > 0$. Therefore, Q is indeed a probability distribution. To show that $H(Q) > \log \frac{N}{2}$ consider $H(Q) - \log \frac{N}{2} = -\sum_{i=1}^{N} q_i \log \left(q_i \frac{N}{2}\right)$. Now if $0 < \delta < \frac{1}{N}$ then $\forall i,\ \ \frac{N}{2} q_i < 1$, implying that $H(Q) > \log \frac{N}{2}$. Since $H(P) \leq \log \frac{N}{2}$, $P \neq Q$. Hence, for each δ we have defined a 1-1 mapping of distributions with entropy below $\log \frac{N}{2}$ to those with entropy above it.

Consequently, the number of distributions with entropy above $\log \frac{N}{2}$ is at least as much as the number of those with entropy below it. This is illustrated using the dotted curve in Figure 2.1 for the case $N = 4$. For the simulations we fixed the resolution and did not distinguish between two probability distributions for which the probability assignments for all data points is within some small range. We then generated all the conditional probability distributions and their (normalized) histogram. This is plotted as the dotted curve superimposed on the bounds in Figure 2.1. It is clearly evident that most of the distributions lie in the high entropy region, where the relation between the entropy and error in Theorem 2.4 carries useful information.

4. Analysis of Classification Error of Estimated (*Mismatched*) Distribution

While in the previous section, we bounded the Bayes optimal error assuming the correct joint probability is known, in this section the more interesting case is investigated – the mismatched probability distribution. The assumption is that the learner has estimated a probability distribution that is different from the true joint distribution and this estimated distribution is then used for classification. The mismatch considered can either be because of the limited number of samples used for learning or because of the assumptions made in the form of the distribution. The effect of the former, decays down with the increasing number of training sample but the effect of the later stays irrespective of the size of the training sample. This work studies the later effect. We assume that we have enough training data to learn the distributions but the mismatch may arise because of the assumptions made in terms of the model. This section studies the degradation of the performance because of this mismatch between the true and the estimated distribution. The performance measure used in our study is the *probability of error.* The problem is being analyzed under two frameworks - group learning or hypothesis testing framework (where one observes a number of samples from a certain class and then makes a decision) and the classification framework (where decision is made independently for each sample.) As is shown, under both of these frameworks, the probability of error is bounded from above by a function of KL-distance between the true and the approximated distribution.

4.1 Hypothesis Testing Framework

Given a sequence $(X_1, ..., X_M)$ of random variables, the area of statistical hypothesis testing attempts to determine whether all the samples (in the given sequence) came from hypothesis H_1 or H_0

$$H_0 \sim P_0(X_1, ..., X_M) \quad \text{and} \quad H_1 \sim P_1(X_1, ..., X_M). \tag{2.25}$$

Likelihood ratio test is the standard way of making decisions in the hypothesis testing framework which is similar to the Bayesian classification paradigm (Eqn 2.4).

This section analyzes the probability of misclassification from the perspective of hypothesis testing. That is, this is the probability of misclassifying a sample as coming from hypothesis $H_0 \sim P_0(X_1, ..., X_M)$ when it actually came from $H_1 \sim P_1(X_1, ..., X_M)$, and vice versa. Since it is assumed that the distributions P_0 and P_1 have already been estimated from data, this perspective (i.e., looking at many samples $X_1, ..., X_M$) allows us to obtain better bounds for the performance of these estimated distributions[3]. This outlook allows one to group the probability of error into two categories α and β: α (Type I error) is the probability of misclassification when the true hypothesis is $H_0 \sim P_0$ and β (Type II error) is the misclassification error when the true hypothesis is $H_1 \sim P_1$. Formally, if $A = \{x : \frac{P_0(x)}{P_1(x)} > \tau\}$ is the acceptance region for hypothesis H_0, then $\alpha = P_0(A^c)$ and $\beta = P_1(A)$. Note that A_M, α_M, and β_M denote the corresponding terms when the decision is made for M random vectors.

Stein's lemma [Cover and Thomas, 1991] gives asymptotic bounds on the performance of a classifier which is using Likelihood ratio test for deciding between the two hypotheses. It shows that under the condition that $\alpha_M < \epsilon$, and for $0 < \epsilon < \frac{1}{2}$, defining $\beta_M^\epsilon = \min_{\alpha_M < \epsilon} \beta_M$ gives

$$\lim_{\epsilon \to 0} \lim_{M \to \infty} \frac{1}{M} \log \beta_M^\epsilon = -D(P_0||P_1). \tag{2.26}$$

In practice, however, rather than the true joint distribution over the samples, the estimated distribution from the data (which may be the induced product distribution, derived using conditional independence assumptions) is used. The result given by Stein's lemma does not hold in this case (i.e. when true distribution is not known) and we prove a modified version of it for this case.

THEOREM 2.9 *(Modified Stein's Lemma) Let $X_1, ..., X_M$ be $i.i.d \sim Q$. Consider the hypothesis test between two hypothesis $Q \sim P_0$, and $Q \sim P_1$. Let*

[3] For the purpose of the analysis of the performance, we study performance using error on the sample.

A_M be the acceptance region for hypothesis $H_0 = Q \sim P_0$. The probabilities of error can then be written as $\alpha_M = P_0^M(A_M^c)$ and $\beta_M = P_1^M(A_M)$. Assume $P_0^{'}$ is used instead of P_0 for the likelihood ratio test. Then if A_M is chosen such that $\alpha_M < \epsilon$, then the type II error (β) is given by

$$\lim_{\epsilon \to 0} \lim_{M \to \infty} \frac{1}{M} \log \beta_M^{\epsilon} = -D_{P_0}(P_0^{'}||P_1) = -E_{P_0}\left(\log \frac{P_0^{'}}{P_1}\right). \tag{2.27}$$

See Appendix A for the proof.

By writing $D_{P_0}(P_0^{'}||P_1)$ in a more recognizable form, the asymptotic bound on the error can be written as:

$$\frac{1}{n}\log(error) \leq -D_{P_0}(P_0^{'}||P_1) = -D(P_0||P_1) + D(P_0||P_0^{'}). \tag{2.28}$$

The first term on the right hand side of Eqn 2.28 is the same as the one in the original Stein's Lemma. Since Stein's lemma gave the minimum error achievable by any algorithm, we cannot do better than this quantity which can be viewed as a "baseline" term. Improving the approximation affects the second term - the distance between the true distribution and the approximation - which acts as the actual penalty.

Although the above bound is derived under the assumption that only the distribution corresponding to one hypothesis is approximated, a similar bound can be derived for the more general case (when the distributions corresponding to both hypothesis are unknown) under the condition that $P_1(x) \leq KP_1^{'}(x)$ for some finite K. In this case, the bound will be given by $\log D_{P_0}(P_0^{'}||P_1^{'})$. The condition is fairly general and always holds for product distributions. However, the bound given by Eqn 2.27 highlights some basic properties of the distributions and will be analyzed in the rest of the chapter. The general case follows similar arguments. Eqn 2.28 shows that the additional penalty term is related to $D(P_0||P_0^{'})$, with P_0 being the true distribution and $P_0^{'}$ the approximation. In the special case when both P_1 and $P_0^{'}$ are product form distributions, we have:

$$\begin{aligned} D_{P_0}(P_0^{'}||P_1) &= \sum_x P_0(x) \log \frac{P_0^{'}(x)}{P_1(x)} \\ &= \sum_{x^1, x^2, \ldots, x^n} P_0(x^1, x^2, \ldots, x^n) \sum_i \log \frac{P_0(x^i)}{P_1(x^i)} \\ &= \sum_i P_0(x^i) \log \frac{P_0(x^i)}{P_1(x^i)} = D(P_0^{'}||P_1). \end{aligned} \tag{2.29}$$

Corollary 2.10 *If both $P_0^{'}$ and P_1 are product distributions then $\frac{1}{n}\log(error) \leq -D(P_0^{'}||P_1)$, i.e. the bound is independent of the joint distribution and depends just on the marginals.*

4.2 Classification Framework

This section analyzes the effect of the mismatched probability distributions in the standard classification framework where the decision is made independently for each test example. We make use of the results given in [Devroye et al., 1996]. Under the assumption of uniform class probability ($P(y=0) = P(y=1) = 1/2$) and if instead of using the true probability distribution $P_1(x)$ and $P_0(x)$ one decides to use $P'_1(x)$ and $P'_0(x)$, respectively, such that $P_1(x) + P_0(x) = P'_1(x) + P'_0(x)$ (which essentially means that we are assuming that in both cases $P(x)$ remains the same) then we have the following lemma:

LEMMA 2.11 *The classification error probability, when estimated distributions (as defined above) are used, is bounded from above by:*

$$\hat{\epsilon} \leq \epsilon + \sqrt{2D(P_1||P'_1)} \tag{2.30}$$

where $\hat{\epsilon}$ is the error that one makes by using the approximate distribution and ϵ is the Bayes optimal error.

Proof. The proof is based on the lemma given in ([Devroye et al., 1996], p. 16). Let $g(x) = 1$ whenever $P_1(x) > P_0(x)$ and $g'(x) = 1$ whenever $P'_1(x) > P'_0(x)$. Then $\epsilon = P(g(x) \neq y)$ and $\hat{\epsilon} = P(g'(x) \neq y)$. Let I denote an indicator function, which is 1 when its argument is true, else is zero. Then,

$$\begin{aligned}
\hat{\epsilon} - \epsilon &= P(g'(x) \neq y) - P(g(x) \neq y) \\
&= \sum_{x \in \mathcal{X}} P(x)(P(g'(x) \neq y|X=x) - P(g(x) \neq y|X=x)) \\
&= \sum_{x \in \mathcal{X}} P(x)(P(y=1|X=x)(I_{\{g(x)=1\}} - I_{\{g'(x)=1\}}) + \\
&\qquad P(y=0|X=x)(I_{\{g(x)=0\}} - I_{\{g'(x)=0\}})) \\
&= \sum_{x \in \mathcal{X}} P(x)(2P(y=1|X=x) - 1)(I_{\{g(x)=1\}} - I_{\{g'(x)=1\}})) \\
&= \sum_{x \in \mathcal{X}} P(x)(2|P(y=1|X=x) - 1/2)|)I_{\{g(x) \neq g'(x)\}} \\
&\leq \sum_{x \in \mathcal{X}} P(x)(2|P(y=1|X=x) - P'(y=1|X=x)|) \\
&= \sum_{x \in \mathcal{X}} (|P(X=x|y=1) - P'(X=x|y=1)|) \\
&\leq \sqrt{2D(P_1||P'_1)}
\end{aligned}$$

where the first inequality follows from the fact that whenever $I_{\{g(x) \neq g'(x)\}} = 1$, $|P(y=1|X=x) - 1/2)| < |P(y=1|X=x) - P'(y=1|X=x)|$ and the last inequality follows from the fact that L_1 norm is bounded by the function of the KL distance ([Kullback, 1968]) as $L_1(P,Q) \leq \sqrt{2D(P||Q)}$.

Interestingly, both in case of hypothesis testing and standard classification problem, we observe that the additional penalty, when mismatched distributions are used, is directly proportional to the KL distance between the true and the approximate (mismatched) distribution. In the next section, the space from which distributions are sampled is analyzed.

5. Density of Distributions

As noted before, maximum likelihood classification is often done by using a sample $S = \{x_1, , \ldots x_m\} \subset \mathcal{X} = \{0,1\}^n$ to first estimate product distributions over $\mathcal{X}$ (see Eqn 2.8). In this section, we investigate the effect of this estimation (working with product distribution instead of the full joint distribution) on the classification accuracy.

Let P_m be a product distribution induced by the sample S, and consider the set $\mathcal{S}$ of all samples of size m over $\mathcal{X}$ that give rise to the same product distribution P_m. For $S \in \mathcal{S}$, let P_S be the joint distribution over $\mathcal{X}$ induced by S. The main theorem of this section shows that of the number of samples S that share the same product distribution, goes down exponentially fast in their KL distance from the product distribution.

THEOREM 2.12 *Let $S^{\leq\varepsilon}$ be the set of samples in $\mathcal{S}$ for which $D(P_S||P_m) \leq \varepsilon$. Then, for some polynomial $r'(\cdot)$,*

$$\frac{|S^{\leq\varepsilon}|}{|\mathcal{S}|} \geq \frac{1}{r'(m)}(1 - 2^{-m\varepsilon}). \tag{2.31}$$

The next two lemmas provide the main results used in developing the proof. Following the bound given in Theorem 12.1.3 in [Cover and Thomas, 1991] we obtain the following lemma.

LEMMA 2.13 *Let $\mathcal{S}$ be the set of all samples of size m over $\mathcal{X}$ that induce the product distribution P_m. Then, for some polynomial $r(\cdot)$,*

$$\frac{1}{r(m)} 2^{mH(P_m)} \leq |\mathcal{S}| \leq r(m) 2^{mH(P_m)}. \tag{2.32}$$

Notice that there is an exact expression for $|\mathcal{S}|$, given by,

$$|\mathcal{S}| = \prod_{i=1}^{n} \binom{m}{mP^i(x_i),\ m(1 - P^i(x_i))}.$$

However, we are interested in the relation to entropy and thus to classification error and therefore need a bound for this term in terms of the entropy.

LEMMA 2.14 *For any $0 \leq \varepsilon \leq H(P_m)$, let $\mathcal{S}^{\varepsilon} \subseteq \mathcal{S}$ be the set of all samples in $\mathcal{S}$, for which $D(P_S || P_m) = \varepsilon$. Then, for some polynomial $r(\cdot)$,*

$$\frac{1}{r(m)} 2^{m(H(P_m)-\varepsilon)} \leq |\mathcal{S}^{\varepsilon}| \leq r(m) 2^{m(H(P_m)-\varepsilon)}. \tag{2.33}$$

Proof. The main step in the proof of above lemma is given by the following claim:

CLAIM 1 *Let P_m be a product distribution defined by the marginals P^i over $\mathcal{X}$. Then for any joint probability distribution P with the marginals P^i,*

$$H(P) = H(P_m) - D(P||P_m).$$

To see the claim, we observe that P_m is the product distribution induced by P_s, and

$$\begin{aligned}
D(P||P_m) &= \sum_{x \in \mathcal{X}} P(x) \log P(x) - \sum_{x} P(x) \log P_m(x) \\
&= -H(P) - \sum_{x \in \mathcal{X}} P(x^1, \dots x^n) \log P_m(x^1, \dots x^n) \\
&= -H(P) - \sum_{x \in \mathcal{X}} P(x^1, \dots x^n) \log \prod_i P^i(x^i) \\
&= -H(P) - \sum_{x \in \mathcal{X}} P(x^1, \dots x^n) \sum_i \log P^i(x^i) \\
&= -H(P) - \sum_i \sum_{(x^1, \dots x^n) \in \mathcal{X}} P(x^1, \dots x^n) \log P^i(x^i) \\
&= -H(P) - \sum_i \sum_{x^i} \sum_{(x^1, \dots x^{i-1}, x^{i+1}, \dots x^n) \in \mathcal{X}} P(x^1, \dots x^n) \log P^i(x^i) \\
&= -H(P) - \sum_i \sum_{x^i} \{\log P^i(x^i)\} \sum_{(x^1, \dots x^{i-1}, x^{i+1}, \dots x^n) \in \mathcal{X}} P(x^1, \dots x^n) \\
&= -H(P) - \sum_i \sum_{x^i} \{\log P^i(x^i)\} P^i(x^i) \\
&= -H(P) + \sum_i H(P^i) \\
&= -H(P) + H(P_m)
\end{aligned}$$

where the last equality is due to the fact that

$$\sum_i H(P^i) = H(P_m),$$

as P_m is the product distribution over the marginals P^i ([Cover and Thomas, 1991], Theorem 2.6.6). The lemma follows by using $P = P_S$ and $D(P_S||P_m) = \varepsilon$.

From Lemma 2.14 we get that

$$\frac{|\mathcal{S}^\varepsilon|}{|\mathcal{S}|} \geq \frac{1}{r^2(m)} 2^{-m\varepsilon}, \tag{2.34}$$

and by integrating over the range $[0, \varepsilon]$ we get Theorem 2.12.

5.1 Distributional Density

In the previous section, Theorem 2.12 was phrased in terms of the number of sample sets that share the same marginal distribution and thus yield the same classifier. We now prove a similar result directly in terms of the number of joint distributions at a certain distance from their induced product distribution. We assume a fixed resolution τ for the representation of real numbers; two real numbers are indistinguishable if their difference is smaller than τ.

We first prove the results for distribution over an alphabet of size 2.

THEOREM 2.15 *Let P_m be a product distribution over $\mathcal{X} \times \mathcal{X}$, and let $\mathcal{P}$ be the collection of all joint probability P over $\mathcal{X} \times \mathcal{X}$ that induce P_m and $\mathcal{P}^{\geq\varepsilon}$ the subset of $\mathcal{P}$ for which $D(P||P_m) \geq \varepsilon$. Then, for some constants A and B,*

$$\frac{|\mathcal{P}^{\geq\varepsilon}|}{|\mathcal{P}|} \leq A \exp -\sqrt{B\varepsilon}. \tag{2.35}$$

We prove the theorem using the following two lemmas.

LEMMA 2.16 *Let P_m be a product distribution over $\mathcal{X} \times \mathcal{X}$ defined by $p_a = P(X^1 = 1)$ and $p_b = P(X^2 = 1)$. Then there is a 1-1 correspondence between distributions in $\mathcal{P}$ and $\delta \in [p_a p_b - min(p_a, p_b), p_a p_b]$. The mapping is given by*

$$P = [p_a p_b - \delta, p_a(1 - p_b) + \delta, (1 - p_a)p_b + \delta, (1 - p_a)(1 - p_b) - \delta]. \tag{2.36}$$

Proof. The probability distribution P_m is given by

$$P_m = [p_a p_b, p_a(1 - p_b), (1 - p_a)p_b, (1 - p_a)(1 - p_b)].$$

Consider a joint probability distribution $P \in \mathcal{P}$ over $\mathcal{X} \times \mathcal{X}$ with marginals $p_a = P(X^1 = 1)$ and $p_b = P(X^2 = 1)$. Define $\delta = P_{11} - p_a p_b$. By simple algebra we get

$$\begin{aligned} P &= [P_{11}\ P_{10}\ P_{01}\ P_{00}] \\ &= [p_a p_b - \delta, p_a(1 - p_b) + \delta, (1 - p_a)p_b + \delta, (1 - p_a)(1 - p_b) - \delta]. \end{aligned} \tag{2.37}$$

The positivity constraint forces δ to be in the range

$$p_a p_b - min(p_a, p_b) \leq \delta \leq p_a p_b.$$

Therefore, any joint distribution in $\mathcal{P}$ can be expressed as in Eq. 2.36, and every δ in this range defines a unique joint probability distribution in $\mathcal{P}$.

LEMMA 2.17 *Let $\mathcal{P}^{\leq \varepsilon}$ be the subset of $\mathcal{P}$ for which $D(P||P_m) \leq \varepsilon$. Then there is a* 1-1 *correspondence between distributions in $\mathcal{P}^{\geq \varepsilon}$ and $\delta \in I$, where*

$$\begin{aligned} I \ &\subseteq \ \left[p_a p_b - min(p_a, p_b), p_a p_b - min(p_a, p_b) + A/2 \exp(-\sqrt{B\varepsilon})\right] \\ &\cup \left[p_a p_b - A/2 \exp(-\sqrt{B\varepsilon}), p_a p_b\right]. \end{aligned}$$

The mapping is the same as in Eqn. 2.36.

Proof. Given the correspondence between values of δ and distributions in $\mathcal{P}$, one can plot $H(P)$ as a function of δ. For distributions in $\mathcal{P}^{\geq \varepsilon}$,

$$H(P) = H(P_m) - D(P||P_m) \leq H(P_m) - \varepsilon. \tag{2.38}$$

We now show that this upper bound on $H(P)$ implies that δ needs to range in the claimed intervals.

Define

$$\begin{aligned} \beta \ &= \ \sum_{i=1}^{4} p_i^2 = (p_a p_b - \delta)^2 + ((1-p_a)p_b + \delta)^2 \\ &+ \ (p_a(1-p_b) + \delta)^2 + ((1-p_a)(1-p_b) - \delta)^2. \end{aligned} \tag{2.39}$$

Using inequalities 2.15 and 2.14 we get that:

$$1 + \log k - k\beta \leq -\log \beta \leq H(P).$$

Define $h(\delta) = H(P)$, where P is the distribution corresponding to δ in the mapping Eqn 2.36 and define $f(\delta) = 1 + \log k - k\beta$, where β is defined by the distribution corresponding to δ in the mapping Eqn 2.36, as in Eqn 2.38.

To argue for the restricted range of δ we discuss separately the region in which the function f is monotonically increasing and the region in which it is monotonically decreasing. In the first case, in order to prove the lemma we need to show that if, for some constant r, $H(P) \leq r$ then there is a value δ_r such that $h^{-1}(H(P)) \leq \delta_r$. Indeed, given r, choose $\delta_r = f^{-1}(r)$. Then, from the above inequality,

$$r = f(\delta_r) \leq h(\delta_r).$$

Now, for P with $H(P) \leq r$, we have that

$$h^{-1}(H(P)) \leq \delta_r.$$

Otherwise,

$$H(P) \geq f(h^{-1}(H(P))) \geq f(\delta_r) = r,$$

where the left inequality is due to the fact that f is a lower bound of h and the second is because we are working in the region in which f is monotonically increasing. The contradiction proves the lemma.

The next step is to compute the value of δ_r for a particular value of r. Using Eqn 2.38, we see that δ is a parabola in $r = f(\delta_r)$. On solving this yields

$$\delta_1 = \frac{B}{8} - \sqrt{A - \frac{r}{4K}} \quad \text{and} \quad \delta_2 = \frac{B}{8} + \sqrt{A - \frac{r}{4K}} \tag{2.40}$$

where A and B are some functions of p_a and p_b (for our analysis, the actual values of A and B are not important). Note that Eqn 2.38 is the equation of a parabola, which achieves its minimum at $\delta = B/8$. This implies that if we choose a probability distribution such that the corresponding δ is such that $\delta_1 < \delta < \delta_2$, then $f(\delta) \geq r$. At the same time $H(P) > f(\delta)$, implies that for this distribution, $H(P) > r$. Therefore, if a probability distribution is chosen such that $H(P) < r$, then the corresponding δ is either $\delta < \delta_1$ or $\delta > \delta_2$.

As shown earlier, the independence assumption, corresponds to

$$H(P) = H(P_m) - D(P||P_m) = H(P_m) - \epsilon = h - \epsilon = r.$$

Substituting this we get,

$$\delta_1 = \frac{B}{8} - \sqrt{\frac{\epsilon}{4K} + A} \quad \text{and} \quad \delta_2 = \frac{B}{8} + \sqrt{\frac{\epsilon}{4K} + A}. \tag{2.41}$$

Therefore if $P \in \mathcal{P}^{\geq \epsilon}$ then

$$\delta \in \left[Mp, Mp + \frac{B}{8} - \sqrt{\frac{\epsilon}{4K} + A}\right] \cup \left[p_a p_b - \frac{B}{8} + \sqrt{\frac{\epsilon}{4K} + A}, p_a p_b\right] \tag{2.42}$$

where $Mp = p_a p_b - min(p_a, p_b)$. Using the fact that $1 + x \leq e^x$, it is easy to see that

$$\frac{B}{8} - \sqrt{\frac{\epsilon}{4K} + A} < \frac{B}{8} \exp\left(-\frac{8}{B}\sqrt{\frac{\epsilon}{4K} + A}\right).$$

Substituting this in the above equation proves the theorem.

This theorem shows that the number of distributions for which the KL-distance from the product distribution is large, is small and it decays exponentially with the distance from the product distribution. In the existing theorem, we have assumed a uniform prior (in terms of counting argument) over the parameter δ. However, it is argued that since uniform prior is not invariant to re-parameterization, one should use a more non-informative prior and once such prior is the Jeffrey's Prior [Balasubramanian et al., 2000]. Let $P(y|\theta)$ be some probability distribution with parameter θ, then the most non-informative prior on θ is given by

$$\pi(\theta) \propto \sqrt{-\sum_y P(y|\theta)\frac{\partial^2}{\partial\theta^2}\log P(y|\theta)}. \tag{2.43}$$

We argue that even under this prior, similar behavior, as seen in the previous theorem, is observed. However, this analysis is a bit more complicated and we are going to sketch the steps that can be used to derive the same result. We will start by computing the prior distribution over δ (Eqn 2.36.) Therefore, we can write:

$$\begin{aligned}\pi(\delta) &\propto \sqrt{-\sum_x P(x|\delta)\frac{\partial^2}{\partial\delta^2}\log P(x|\delta)}\\ &= \sqrt{\frac{1}{p_ap_b-\delta}+\frac{1}{(1-p_a)(1-p_b)-\delta}+\frac{1}{p_a(1-p_b)+\delta}+\frac{1}{(1-p_a)p_b+\delta}}.\end{aligned} \tag{2.44}$$

In the analysis done earlier, it was assumed that the number of distributions is proportional to the length of the interval in which δ belongs. However, under a given prior distribution over this interval, one needs to integrate the prior distribution over this interval. from $min(p_a, p_b)$ to $\sqrt{\beta - A}$. Lets denote this by p_w^J. Using the Eqn 2.44, we obtain (for some normalization constant C, and without loss of generality, assuming that $p_a < p_b$, and for simplicity of analysis, we assume that values of p_a, p_b are such that we have symmetry, implying that the size of the region from $p_ap_b - p_a$ to δ_1 is same as the size of the region from δ_2 to p_ap_b):

$$\begin{aligned}p_w^J &= C\int_{p_ap_b-p_a}^{\delta_1}\sqrt{\frac{1}{p_ap_b-\delta}+\frac{1}{(1-p_a)(1-p_b)-\delta}+\frac{1}{p_a(1-p_b)+\delta}+\frac{1}{(1-p_a)p_b+\delta}}d\delta\\ &\leq C\int_{p_ap_b-p_a}^{\delta_1}\sqrt{\frac{1}{p_ap_b-\delta}}+\sqrt{\frac{1}{(1-p_a)(1-p_b)-\delta}}+\sqrt{\frac{1}{p_a(1-p_b)+\delta}}+\sqrt{\frac{1}{(1-p_a)p_b+\delta}}d\delta\end{aligned}$$

where we use the fact that $\sqrt{a+b} < \sqrt{a}+\sqrt{b}, \forall a > 0$ and $b > 0$. Now solving the integral, and making use of the fact that $\sqrt{a-b} > \sqrt{a} - \sqrt{b}, \forall a > b > 0$, we obtain

$$p_w^J \leq C_1(C_2 + \sqrt{\delta_1}) \tag{2.45}$$

for some constants C_1, C_2. Substituting δ_1 by result from Eqn 2.40, we obtain a result similar to the one in Eqn 2.42, except for different constants.

This theorem can be easily extended to the case of any number of variables. Consider any joint distribution, the number of terms are 2^n. The number of free variables, once the marginal distribution is fixed are $2^n - n - 1$ (1 comes from the fact that the probabilities have to sum to 1). Let us denote these terms by $\delta_i, \quad 1 \leq i \leq 2^n - n - 1$. Now we can represent the other remaining joint probabilities, in terms of the marginal probabilities and these parameters.

It is straightforward to see that all the remaining joint probabilities can be represented in terms of the linear function of the marginal probabilities and these parameters (δ_i). This implies that on fixing all but δ_j for some j, the problem reduces to the case discussed in the previous theorem. And, from previous theorem, we know that for any fixed probability distribution, if we vary one parameter, then the number of distributions which are far from the product distribution decays exponentially.

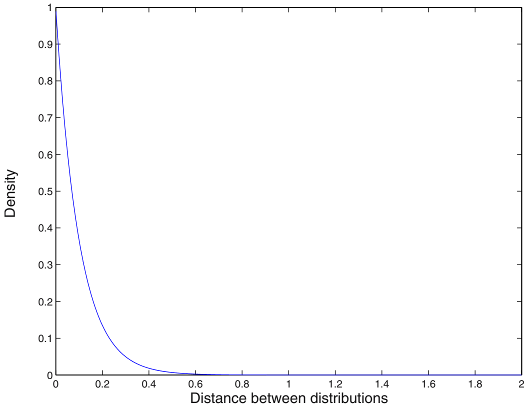
5.2 Relating to Classification Error

Together with the penalty results in Sec. 2.4, it is clear as to why we represent the distributional density in terms of the distance between the distributions. If, as probabilistic classifier do, classification with respect to P is done using the induced product distribution P_m, then the error incurred is related to $D(P||P_m)$ (disregarding the baseline term in Eqn 2.28). Therefore, Theorem 2.12 implies that for most data sets, the classification error is going to be small.

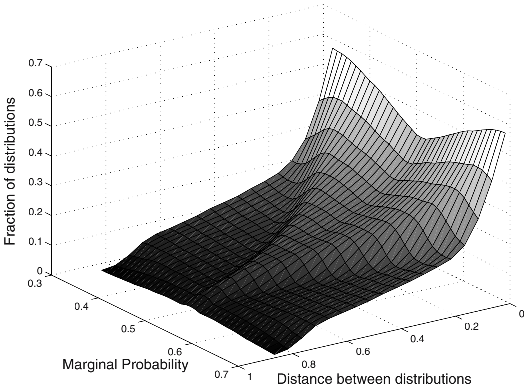
In Figure 2.2, the plotted histogram shows the density of the joint distributions which have the same marginal distribution, as a function of the product distribution and the distance between the joint and the product distribution ($D(P||P_m)$[4]). For example, consider two random variables $x^1, x^2 \in \{0,1\}$. Lets fix $P(x^1 = 1) = 0.8$ and $P(x^2 = 1) = 0.2$ (i.e. fixing the marginal distribution). This means that $P(x^1 = 1, x^2 = 1)$ can take only finite number of values (if we limit the resolution of the probabilities to say 0.001). Thus, the figure shows that the "bad" cases (when the distribution is far from marginal) are rare when considering the space of all possible distributions with a given marginal distribution (or all data sets sampled according to distributions with a given marginal). Note that, this is an upper bound analysis. Sometimes this bound is tight, as shown in Sec. 2.4 for the case in which P_1 is in product form. Nevertheless, there could be cases in which the bound is loose. However, the important point to note is that the bound goes in the right direction, and in the majority of the cases the upper bound is small.

Fig. 2.3, gives the results of simulations to validate the results presented in the chapter. We considered a case of 2 and 3 features as input and the case of a binary classifier. In each case, 1000 sequences of fixed length were randomly sampled according to different joint distributions, all having the same induced

[4] Notice that this distance is always finite since P_m is 0 iff P is zero.

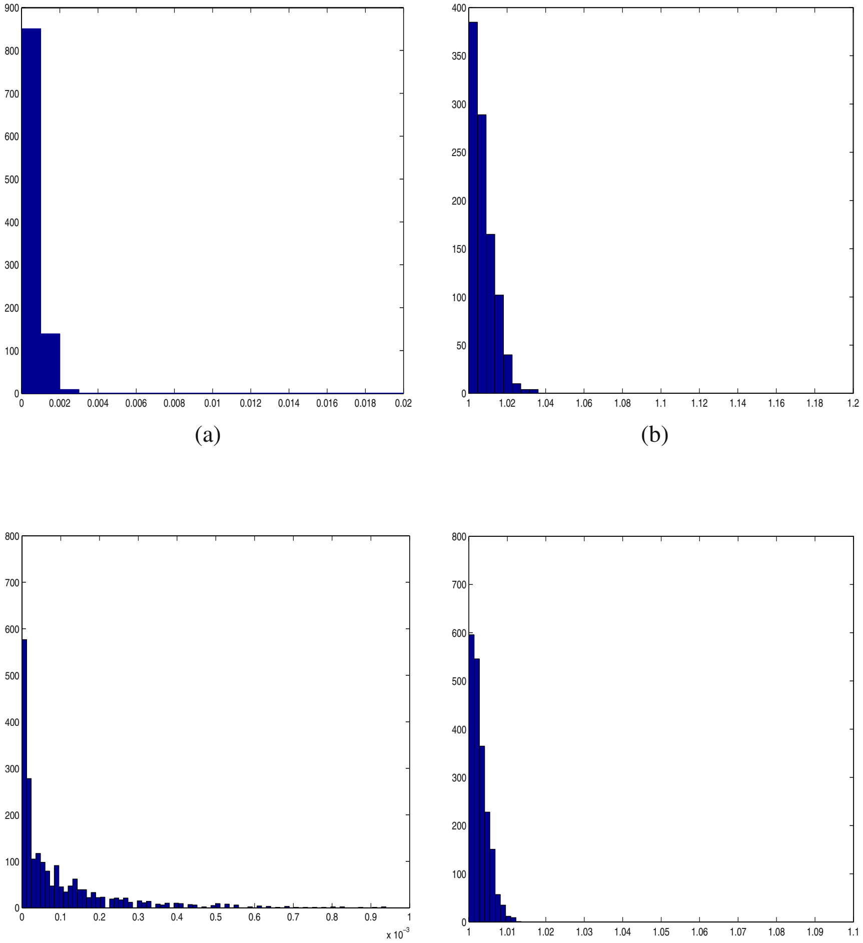


(a)



(b)

Figure 2.2. (a) Density of sequences of length n. The Y-axis gives the number of sequences ($K2^{-n\epsilon}$) as a function of the distance of the true joint distribution from the product distribution ($D(P_0||P_{m0}) = \epsilon$) in the X-axis. (b) shows the decay in the number of the distributions as a function of the entropy of the marginal distribution and the distance of the joint distribution and its induced product distribution. Plots are based on a two attributes case. P_m varies from [0.3 0.3] to [0.7 0.7] (i.e., the attributes have the same marginal distribution).



(a) (b)

(c) (d)

Figure 2.3. The plots (a) and (b) show histograms of the number of sample sets with a given product distribution as a function of the KL distance between the joint distribution of the sample set and the product distribution. Plots (c) and (d) give the ratio between the classification error when using the product distribution and the Bayes optimal error (modeling complete joint distribution).

product distribution. Plots of the number of sequences, with a joint distribution at a certain distance from the product distribution are given in Fig. 2.3 (a) and (c) (for 2 and 3 features respectively). As expected, the histogram looks very similar to the one in Fig. 2.2. We also show in Fig. 2.3 (b) and (d) for 2 and 3 features, respectively, the resulting classification errors as a function of the distance between the joint and the product distribution. The figures give the ratio of the errors made during classification when one uses the product distribution vs. the use of the true joint distribution. As expected the error ratio ($exp(-D(P||P_m))$) has an exponential decay.

6. Complex Probabilistic Models and Small Sample Effects

In the practice of machine learning [Golding, 1995; Schneiderman and Kanade, 2000], the use of probabilistic classification algorithms is preceded by the generation of new features from the original attributes in the space which can be seen as using complex probabilistic classifiers. We analyze the particular case of Tree-Augmented Naive Bayes (TAN) classifier introduced in [Friedman et al., 1997], which is a sophisticated form of the Naive Bayes classifier modeling higher (second) order probabilistic dependencies between the attributes. More details on the TAN classifier are given in Chapter 7.

Table 2.1. This table compares the performance of Naive Bayes classifier (NB) with the Tree-Augmented Naive Bayes classifier (TAN). The results presented here are the ones published in [Friedman et al., 1997]. The Avr. Diff. column is the average (over the two classes) of the distances between the TAN and the Naive product distributions.

Dataset	$D(P_0\|\|P_{m0})$	$D(P_1\|\|P_{m1})$	$D(P_0\|\|P_1)$	$D(P_1\|\|P_0)$
Pima	0.0957	0.0226	0.9432	0.8105
Breast	0.1719	0.4458	6.78	9.70
Mofn-3-7-10	0.3091	0.3711	0.1096	0.1137
Diabetes	0.0228	0.0953	0.7975	0.9421
Flare	0.5512	0.7032	0.8056	0.8664

Dataset	Avg. Diff	NB Res	TAN Res
Pima	0.8177	75.51±1.63	75.13±1.36
Breast	7.9311	97.36±0.50	*95.75±1.25*
Mofn-3-7-10	-0.2284	86.43±1.07	**91.70±0.86**
Diabetes	0.8108	74.48±0.89	75.13±0.98
Flare	0.2088	79.46±1.11	**82.74±1.60**

Friedman et al. [Friedman et al., 1997] have conducted a number of experiments and reported improved results on some of the datasets by the use of TAN over the standard naive Bayes. It is easy to see that by modeling the TAN distribution, one is essentially decreasing the distance between the true (joint) and the approximated distribution. i.e. $D(P||P_m) \geq D(P||P_{TAN})$ where P_{TAN} refers to the probability distribution modeled by TAN. The proof of this is given in Appendix B. Replacing P by either P_0 or P_1 reduces to the case presented in Section 2.4. Based on the results developed in the previous sections, one can argue that reduction in $D(P||P_m)$ is directly mapped to the reduction in the bound on error, thus explaining the better performance. Table 2.1 exhibits this result when evaluated on five data sets (chosen based on the number of attributes and training examples) studied in [Friedman et al., 1997]. In addition to presenting the results published in [Friedman et al., 1997], we have computed, for each one of the classes (0, 1), the distance between the pure naive Bayes and the TAN distribution, and their average. The Avr. Diff. column is the average (over the two classes) of the distances between the TAN and the product distributions. Clearly our results predict well the success (rows 3, 5) and failure (row 2) of TAN over the Naive Bayesian distribution.

As mentioned before, in this chapter we have ignored small sample effects, and assumed that good estimates of the statistics required by the classifier can be obtained. In general, when the amount of data available is small, the naive Bayes classifier may actually do better than the more complex probability models because of the insufficient amount of data that is available. In fact, this has been empirically observed and discussed by a number of researchers [Friedman, 1997; Friedman et al., 1997].

7. Summary

In the last few years we have seen a surge in learning work that is based on probabilistic classifiers. While this paradigm has been shown experimentally successful on many real world applications, it clearly makes vastly simplified probabilistic assumptions. This chapter uses an information theoretic framework to resolve the fundamental question of: why do these approaches work. On the way to resolving this puzzle, we developed methods for analyzing probabilistic classifiers and contributed to understanding the tradeoffs in developing probabilistic classifiers and thus to the development of better classifiers.

However, another view point in understanding this theory comes from the probably approximately correct (PAC) viewpoint. In the next chapter, we use the PAC framework and try to understand why learning works in many cognitive situations which otherwise seem to be very hard to learn.

Appendix A

THEOREM 2.9 (Modified Stein's Lemma) *Let $X_1, ..., X_M$ be $i.i.d \sim Q$. Consider the hypothesis test between two hypothesis $Q \sim P_0$, and $Q \sim P_1$. Let A_M be the acceptance region for hypothesis $H_0 = Q \sim P_0$. The probabilities* *of error can be written as*

$$\alpha_M = P_0^M(A_M^c) \quad and \quad \beta_M = P_1^M(A_M). \tag{2.46}$$

Assume $P_0^{'}$ is used instead of P_0 for the likelihood ratio test. Then if A_M is chosen (under the likelihood ratio test, see Eqn 2.48) such that $\alpha_M < \epsilon$, then the type II error (β) is given by

$$\lim_{\epsilon \to 0} \lim_{M \to \infty} \frac{1}{M} \log \beta_M^{\epsilon} = -D_{P_0}(P_0^{'}||P_1) = -E_{P_0}(\log \frac{P_0^{'}}{P_1}) \tag{2.47}$$

Proof. Let us define A_M as

$$A_M = \left\{ x \in \mathcal{X}^M : 2^{M(D_{P_0}(P_0^{'}||P_1)-\delta)} \leq \frac{P_0^{'}(x)}{P_1(x)} \leq 2^{M(D_{P_0}(P_0^{'}||P_1)+\delta)} \right\}.$$

First we will show that with this definition indeed $\alpha_M < \epsilon$; this is done by showing that $1 - \alpha_M = P(A_M) \to 1$. The second step will prove the statement made in Eqn 2.27.

1. $\lim\limits_{M \to \infty} P_0(A_M) \to 1$. This follows from

$$\begin{aligned} \lim_{M \to \infty} P_0(A_M) &= \lim_{M \to \infty} P_0^M \left\{ \frac{1}{M} \sum_{i=1}^{M} \log \frac{P_0^{'}(x_i)}{P_1(x_i)} \right. \\ &\in \left. \Big(D_{P_0}(P_0^{'}||P_1) - \delta, D_{P_0}(P_0^{'}||P_1) + \delta \Big) \right\} \to 1 \end{aligned}$$

which follows directly from the strong law of large numbers.

2. Using the definition of A_M, we have

$$\begin{aligned} P_1^n(A_M) = \sum_{A_M} P_1(x) &\leq \sum_{A_M} P_0^{'}(x) 2^{-M(D_{P_0}(P_0^{'}||P_1)-\delta)} \\ &= 2^{-M(D_{P_0}(P_0^{'}||P_1)-\delta)} K \end{aligned}$$

where $K = \sum\limits_{A_M} P_0^{'}(x)$, a constant less than 1.

Similarly, it is straight forward to show that

$$P_1^M(A_M) \geq 2^{-M(D_{P_0}(P_0^{'}||P_1)+\delta)} K.$$

Now if we denote $\beta_M = P_1^M(A_M)$, the error of type II, then

$$-D_{P_0}(P_0^{'}||P_1) - \delta + \frac{\log K}{M} \leq \frac{1}{M} \log \beta_M \leq -D_{P_0}(P_0^{'}||P_1) + \delta + \frac{\log K}{M}.$$

Therefore, it follows that

$$\lim_{M \to \infty} \frac{1}{M} \log \beta_M = -D_{P_0}(P_0^{'}||P_1).$$

Appendix B

Our analysis presented in the following theorem justifies the improvement in results obtained by Tree-Augmented Naive Bayes networks over the simple naive Bayes classifier.

THEOREM 2.18 *Let P be the true join probability distribution over the random variables $x^1, x^2, ..., x^M$. Let P_m be the induced product distribution (under the assumption that all the random variables are independent of each other). Let P_{TAN} be the induced product distribution over some pairs or random variables and which considers the other variables to be independent of each other. To simplify the analysis, we will assume that P_{TAN} models the joint distribution of x^1, x^2 and considers all other variables to be independent of each other. Then,*

$$D(P||P_m) \geq D(P||P_{TAN}). \tag{2.48}$$

Proof. Under the assumption

$$\begin{aligned} D(P||P_m) - D(P||P_{TAN}) &= \sum_{x \in \mathcal{X}} P \log \frac{P}{P_m} - \sum_{x \in \mathcal{X}} P \log \frac{P}{P_{TAN}} \\ &= \sum_{x \in \mathcal{X}} P \log \frac{P_{TAN}}{P_m} = \sum_{x \in \mathcal{X}} P(x^1, x^2) \log \frac{P(x^1, x^2)}{P(x^1)P(x^2)} \geq 0. \end{aligned}$$

The last inequality follows from the fact the Kullback-Leibler divergence is always positive.

Chapter 3

THEORY: GENERALIZATION BOUNDS

Motivated by the analysis of the coherency constraints done in the previous chapter, we further extend the understanding of learning algorithms by developing a data dependent approach which is used to derive generalization bounds that depend on the margin distribution.

This chapter develops a learning theory that is relevant for learning in very high dimensional spaces and uses it to establish informative generalization bounds, even for very high dimensional learning problems. The approach is motivated by recent works [Roth and Zelenko, 2000; Garg and Roth, 2001a; Arriaga and Vempala, 1999] that argue that some high dimensional learning problems are naturally constrained in ways that make them, effectively, low dimensional problems. In these cases, although learning is done in a high dimension, generalization ought to depend on the true, lower dimensionality of the problem.

In this chapter, we present our method, we analyzes it, and we use it to develop new generalization bounds. We then evaluate the projection profile based method experimentally on real data and show its effectiveness. Specifically, we show that in many of the high dimensional problems (e.g. in the natural language domain), our bound is tighter than the Vapnik-Chervonenkis (VC) dimension or the fat-shattering based bound.

1. Introduction

The study of generalization abilities of learning algorithms and its dependence on sample complexity is one of the fundamental research efforts in learning theory. Understanding the inherent difficulty of learning problems allows one to evaluate whether learning is at all possible in certain situations, estimate the degree of confidence in the predictions made by learned classifiers, and is crucial in understanding and analyzing learning algorithms.

Understanding generalization is even more important when learning in very high dimensional spaces, as in many natural language and computer vision applications. Specifically, can one count on the behavior of a 10^6 dimensional classifier that is trained on a few examples, or even a few thousands examples? Existing bounds are loose and essentially meaningless in these (and even in simpler) cases[1].

Technically, our approach builds on recent developments in the area of random projection of high dimensional data [Johnson and Lindenstrauss, 1984] which show, informally, that it is possible to project high dimensional data randomly into a much smaller dimensional space, with a relatively small distortion of the distances between the projected points. This result is extended here to apply in the context of a sample of points along with a linear classifying hypothesis, and is used to develop generalization bounds for linear classifiers in very high dimensional spaces. The basic intuition underlying our results is as follows. If the effective dimensionality of the data is much smaller than the observed dimensionality, then it should be possible to randomly project the data into a lower dimensional space while, due to a small distortion in distances, incurring only a small amount of classification error, relative to what is possible in the original, high dimensional space. Since the projected space has low dimension, better “standard" generalization bounds hold there.

We introduce a new, data dependent, complexity measure for learning. The *projection profile* of data sampled according to a distribution $\mathcal{D}$, is the expected amount of error introduced when a classifier h is randomly projected, along with the data, into k-dimensions. Although this measure seems somewhat elusive, we show that it is captured by the following quantity: $a_k(\mathcal{D}, h) = \int_{x \in \mathcal{D}} u(x) d\mathcal{D}$, where

$$u(x) = \min\left(3 \exp\left(-\frac{(\nu(x))^2 k}{8(2 + |(\nu(x))|)^2}\right), 1\right) \tag{3.1}$$

and $\nu(x)$ is the distance between x and the classifying hyperplane[2] defined by h, a linear classifier for $\mathcal{D}$. The sequence $\mathcal{P}(\mathcal{D}, h) = (a_1(\mathcal{D}, h), a_2(\mathcal{D}, h), \ldots)$ is the *projection profile* of $\mathcal{D}$.

The projection profile turns out to be quite informative, both theoretically and in practice. In particular, it decreases monotonically (as a function of k), and provides a trade-off between dimension and accuracy. Namely, if the data is transformed from n to k dimensions then we expect the amount of error introduced to be $a_k(\mathcal{D}, h)$. This new complexity measure allows us to state

[1] Several statistical methods can be used to ensure the robustness of the empirical error estimate [Kearns et al., 1997]. However, these typically require training on even less data, and do not contribute to understanding generalization and the domain.

[2] Our analysis does not assume the data to be linearly separable.

a generalization bound in terms of the lower-dimensional projected space - the effective dimension of the data. We show that the overall performance will depend on an *estimation* of the projection profile when projecting to the effective dimension, with the addition of the standard, Vapnik-Chervonenkis (VC)-dimension arguments, in the projected space.

Our approach suggests a significant improvement over current approaches to generalization bounds, which are based either on VC theory [Vapnik, 1982] and learning theory versions of Occam's Razor [Blumer et al., 1989] or, more recently, on the *margin* of a classifier with respect to a sample [Shawe-Taylor, 1998; Shawe-Taylor and Cristianini, 1999; Shawe-Taylor and Christianini, 2000; Herbrich and Graepel, 2001].

Although the development of margin-based bounds has been a significant improvement, they still are not meaningful. The main shortcoming is that the margin of the data might be defined by very few points of the distribution and thus might be very small. Our method can be viewed as allowing an explicit dependency on the distribution of the geometric distances of points from the classifier, rather than only the extreme points. We refer to this distance as the *margin distribution* of the data. In our method, the contribution of those nearby "problematic" points to the generalization bound is weighted together with their portion in the distribution. This is significant when most of the data is far from the optimal classifier - only very few points, those that determine the margin, are close to it. Our experiments reveal that this is indeed the case. The advantage of our method is exhibited in Fig. 3.5, showing the margin distribution of data taken from a high dimensional natural language classification problem [Golding and Roth, 1999]. Despite the zero margin in this case, our method provides an informative bound.

2. Preliminaries

We study a binary classification problem $f : \Re^n \rightarrow \{-1, 1\}$. $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ denotes a sample set of m examples. The hypothesis $h \in \Re^n$ is an n-dimensional linear classifier assumed, without losing generality, to pass through the origin. That is, for an example $x \in \Re^n$ the hypothesis predicts $\widehat{y}(x) = sign(h^T x)$. Throughout the chapter, we use n to denote the original (high) dimensionality of the data, k to denote the (smaller) dimension into which projections are made and m to denote the sample size of the data.

DEFINITION 3.1 *The loss function considered is the* 0-1 *loss. Under this loss function, the* empirical error $\widehat{E}$ *of* h *over a sample set* S *and the* expected error $\overline{E}$ *are given resp. by,*

$$\begin{aligned} \widehat{E}(h, S) &= \frac{1}{m} \sum_i I(\widehat{y}(x_i) \neq y_i) \\ \overline{E}(h, S) &= E_x\Big[\widehat{y}(x) \neq f(x)\Big], \end{aligned}$$

where $I(\cdot)$ is the indicator function which is 1 *when its argument is true and* 0 *otherwise. The expectation E_x is taken over the distribution of data.*

We denote by $||\cdot||$ the L_2 norm of a vector. Clearly, we can assume without losing generality that all data points come from the surface of unit sphere (i.e. $\forall x, ||x|| = 1$), and that $||h|| = 1$ (as the classification just depends on the sign of the inner product). Under these assumptions, the classification output $\widehat{y}(x)$ can be interpreted as the sign of the angle between the vectors x and h.

Let $\nu(x) = h^T x$ denote the signed distance of the sample point x from the classifier h. When x refers to the j^{th} sample from the sample set S, we denote it by $\nu_j = \nu(x_j) = h^T x_j$. With this notation (omitting the classifier h) the classification rule reduces simply to $sign(\nu(x))$.

Our method makes use of *random projections*, introduced in [Johnson and Lindenstrauss, 1984] and studied intensively since then [Arriaga and Vempala, 1999; Indyk, 2001].

Definition 3.2 (**Random Matrix**) *Let R be a $k \times n$ matrix where each entry $r_{ij} \sim N(0, 1/k)$. R is called a* random projection matrix. *For $x \in \Re^n$, we denote by $x' = Rx \in \Re^k$ the projection of x from an n to a k-dimensional space using projection matrix R.*

In a similar fashion, for a classifier $h \in \Re^n$, h' will denote its projection to a k dimensional space via R (omitting k when clear from the context). Likewise, S' denotes the set of points which are the projections of the sample S, and $\nu_j^{'} = (h')^T x_j^{'}$, the signed distance of the projected point from the projected classifier.

Briefly, the method of random projection shows that with high probability, when n dimensional data is projected down to a lower dimensional space of dimension k, using a random $k \times n$ matrix, relative distances between points are *almost* preserved. Formally:

Theorem 3.3 ([Arriaga and Vempala, 1999], Theorem 1) *Let $u, v \in \Re^n$ and let u' and v' be the projections of u and v to $\Re^k$ via a random matrix R chosen as described above. Then, for any constant c,*

$$P\left[(1-c) \leq \frac{||u'-v'||^2}{||u-v||^2} \leq (1+c)\right] \geq 1 - e^{-c^2 k/8}, \tag{3.2}$$

where the probability is over the selection of the random matrix R.

Note that if $||u|| = ||v|| = 1$, then $||u - v|| = 2 - 2u \cdot v$. Therefore, the above theorem can also be viewed as stating that, with high probability, random projection preserves the angle between vectors which lie on the unit sphere.

3. A Margin Distribution Based Bound

As mentioned earlier, the decision of the classifier h is based on the sign of $\nu(x) = h^T x$. Since both h and x are normalized, $|\nu(x)|$ can be thought of as the geometric distance between x and the hyperplane orthogonal to h that passes through the origin. Given a distribution on data points x, this induces a distribution on their distance from the hyperplane induced by h, which we refer to as the *margin distribution*.

Note that this is different from the margin of the sample set S with respect to a classifier h. Traditionally in the learning community, margin of a sample set S (referred to simply as "the margin") is the distance of the point which is closest to the hyperplane. Formally, the *margin of S with respect to h* is given by:

$$\gamma(S, h) = \min_{i \in \{1,\ldots,m\}} |h^T x_i|. \tag{3.3}$$

Consider a scenario in which one learns a classifier in a very high dimensional space (typical in image processing, language processing, and data mining applications). According to existing theory, in order to learn a classifier which, with high confidence, performs well on previously unseen data, one needs to train it on a large amount of data. In what follows, we develop alternative bounds that show that if "many" of the high dimensional points are classified with high confidence, that is, $|h^T x|$ is large for these, then one does not need as many points as predicted by VC-theory or margin based theory. The main result of the chapter formalizes this insight and is given in the following theorem.

THEOREM 3.4 *Let $S = \{(x_1, y_1), \ldots, (x_{2m}, y_{2m})\}$ be a set of n-dimensional labeled examples and h a linear classifier. Then, for all constants $0 < \delta < 1; 0 < k$, with probability at least $1 - 4\delta$, the expected error of h is bounded by*

$$\overline{E} \leq \widehat{E}(S, h) + \min_k \left\{ \mu_k + 2\sqrt{\frac{(k+1)\ln\frac{me}{k+1} + \ln\frac{1}{\delta}}{2m}} \right\} \tag{3.4}$$

where $\mu_k = \frac{6}{m\delta} \sum_{j=1}^{2m} \exp\left(-\frac{\nu_j^2 k}{8(2+|\nu_j|)^2}\right)$, and $\nu_j = \nu(x_j) = h^T x_j$.

3.1 Proving the Margin Distribution Bound

The bound given in Eqn 3.4 has two main components. The first component, μ_k, is the distortion incurred by the random projection to dimension k, and the second follows directly from VC theory for this dimension.

Recall that the random projection theorem states that relative distances are (almost) preserved when projecting to lower dimensional space. Therefore, we first argue that the image, under projection, of data points that are far from h in the original space, will still be far from its image in the projected (k dimensional) space. The first term quantifies the penalty incurred due to data points whose images will not be consistent with the image of h. That is, this term bounds the *empirical* error in the projected space. Once the data lies in the lower dimensional space, we can bound the expected error of the classifier on the data as a function of the dimension of the space, number of samples and the empirical error there (that is, the first component).

Decreasing the dimension of the projected space implies increasing the contribution of the first term, while the VC-dimension based term decreases. To get the optimal bound, one has to balance these two quantities and choose the dimension k of the projected space so that the generalization error is minimized. We will use the following lemmas to compute the penalty incurred while projecting the data down to k dimensional space.

LEMMA 3.5 *Let h be an n-dimensional classifier, $x \in \Re^n$ a sample point, such that $||h|| = ||x|| = 1$, and $\nu = h^T x$. Let $R \in \Re^{k \times n}$ be a random projection matrix (Definition 3.2), with $h' = Rh, x' = Rx$. Then, the probability of misclassifying x, relative to its classification in the original space, due to the random projection, is*

$$P\Big[sign(h^T x) \neq sign(h'^T x')\Big] \leq 3\exp\left(-\frac{\nu^2 k}{8(2+|\nu|)^2}\right). \tag{3.5}$$

Proof. From Theorem 3.3 we know that with probability at least

$$Z(c) = 1 - 3\exp\left(-\frac{c^2 k}{8}\right)$$

we have

$$\begin{gathered}(1-c)||h||^2 \leq ||h'||^2 \leq (1+c)||h||^2,\\ (1-c)||x||^2 \leq ||x'||^2 \leq (1+c)||x||^2,\\ (1-c)||h-x||^2 \leq ||h'-x'||^2 \leq (1+c)||h-x||^2.\end{gathered}$$

Since $||h|| = ||x|| = 1$, and setting $\nu = h^T x,\ \nu' = h'^T x'$, we have

$$\begin{aligned}||h-x||^2 &= ||h||^2 + ||x||^2 - 2h^T x = 2 - 2\nu.\\ ||h'-x'||^2 &= ||h'||^2 + ||x'||^2 - 2\nu'.\end{aligned}$$

Which, along with the above inequalities gives,

$$||h'||^2 + ||x'||^2 - 2\nu' \leq (1+c)(2-2\nu).$$

This implies,

$$\begin{aligned}\nu' &\geq \frac{||h'||^2 + ||x'||^2}{2} - (1+c)(1-\nu) \\ &\geq (1-c) - (1+c)(1-\nu) \\ &= c(\nu-2) + \nu.\end{aligned}$$

Thus, when $\nu > 0$ we have $\nu' > 0$ if $c(\nu - 2) + \nu > 0$. This implies that we need $\nu/(2-\nu) > c$.
Similarly,

$$||h'||^2 + ||x'||^2 - 2\nu' \geq (1-c)(2-2\nu).$$

Thus,

$$\begin{aligned}\nu' &\leq \frac{||h'||^2 + ||x'||^2}{2} - (1-c)(1-\nu) \\ &\leq 1 + c - 1 + c + \nu(1-c) \\ &= 2c + \nu(1-c) \\ &= c(2-\nu) + \nu.\end{aligned}$$

In particular, if $\nu < 0$ then $\nu' < 0$ if $c(2-\nu) + \nu < 0$, which implies $c < -\nu/(2-\nu)$.

Combining the above two inequalities, we conclude that ν and ν' have the same sign if $c < |\nu|/(2+|\nu|)$. Namely, the required probability is

$$\begin{aligned}P\Big[sign(h^T x) \neq sign(h'^T x')\Big] &\leq 1 - Z\left(\frac{|\nu|}{2+|\nu|}\right) \\ &= 3\exp\left(-\frac{\nu^2 k}{8(2+|\nu|)^2}\right),\end{aligned}$$

which is obtained by picking $c = |\nu|/(2+|\nu|)$.

Next, we define the projection error for a sample – this is essentially the projection profile introduced in Section 3.1 for a finite sample. A natural interpretation of the projection error is that it is an estimate of the projection profile obtained by sampling the data with respect to a fixed classifier.

DEFINITION 3.6 (**Projection Error**) *Given a classifier h, a sample S, and a random projection matrix R, let $\mathrm{E}rr_{proj}(h, R, S)$ be the classification error caused by the projection matrix R. Namely,*

$$\mathrm{E}rr_{proj}(h, R, S) = \frac{1}{|S|} \sum_{x \in S} I(sign(h^T x) \neq sign(h'^T x')).$$

LEMMA 3.7 *Let h be an n-dimensional classifier, R a random projection matrix, and a sample of m points*

$$S = \Big\{(x_1, y_1), \ldots, (x_m, y_m) \,\Big|\, x_i \in \Re^n, y_i \in \{0,1\}\Big\}.$$

Then, with probability $\geq 1-\delta$ (over the choice of the random projection matrix R), the projection error satisfies $\mathrm{E}rr_{proj}(h, R, S) \leq \varepsilon_1(S, \delta)$, where

$$\varepsilon_1(S, \delta) = \frac{1}{m}\frac{1}{\delta}\sum_{i=1}^{m} 3\exp\left(-\frac{\nu_i^2 k}{8(2+|\nu_i|)^2}\right), \tag{3.6}$$

$\nu_i = h^T x_i$, for $i = 1, \ldots, m$.

Proof. Let Z be the expected projection error of a sample where the expectation is taken with respect to the choice of the projection matrix. That is,

$$\begin{aligned} Z &= E\Big[\mathrm{E}rr_{proj}(h, R, S)\Big] \\ &= E\left[\frac{1}{|S|}\sum_{x\in S} I(sign(h^T x) \neq sign({h'}^T x'))\right] \\ &= \frac{1}{m}\sum_{x\in S} E\Big[I(sign(h^T x) \neq sign({h'}^T x'))\Big] \\ &= \frac{1}{m}\sum_{x\in S} P\Big[sign(h^T x) \neq sign({h'}^T x')\Big] \\ &\leq \frac{1}{m}\sum_{x\in S} 3\exp\left(-\frac{\nu_i^2 k}{8(2+|\nu_i|)^2}\right) = \delta\varepsilon_1(S, \delta), \end{aligned}$$

which follows by linearity of expectation and Lemma 3.5. Now, using Markov inequality,

$$P\left[\mathrm{E}rr_{proj}(h, R, S) \geq \frac{Z}{\delta}\right] \leq \delta,$$

which establishes the lemma, as $Z/\delta \leq \varepsilon_1(S, \delta)$.

LEMMA 3.8 *Let S_1, S_2 be two samples of size m from $\Re^n$, R a random projection matrix, and S_1', S_2' the projected sets. Then, with probability $\geq 1 - 2\delta$,*

$$P\Big[\Big|\widehat{E}(h, S_1) - \widehat{E}(h, S_2)\Big| > \varepsilon\Big] < P\Big[\Big|\widehat{E}(h', S_1') - \widehat{E}(h', S_2')\Big| > \rho\Big],$$

where $\rho = \varepsilon - \varepsilon_1(S_1, \delta) - \varepsilon_1(S_2, \delta)$.

Proof. Applying the result of the Lemma 3.7 on sample sets S_1, S_2, we obtain that with probability at least $1 - 2\delta$, we have

$$\begin{aligned} \left|\widehat{E}(h, S_1) - \widehat{E}(h', S_1')\right| &< \varepsilon_1(S_1, \delta), \quad \text{and} \\ \left|\widehat{E}(h, S_2) - \widehat{E}(h', S_2')\right| &< \varepsilon_1(S_2, \delta). \end{aligned}$$

Using simple algebra, we obtain,

$$\begin{aligned} \left|\widehat{E}(h, S_1) - \widehat{E}(h, S_2)\right| &\leq \left|\widehat{E}(h, S_1) - \widehat{E}(h', S_1')\right| \left|\widehat{E}(h', S_1') - \widehat{E}(h', S_2')\right| \\ &\qquad + \left|\widehat{E}(h', S_2') - \widehat{E}(h, S_2)\right| \\ &\leq \varepsilon_1(S_1, \delta) + \varepsilon_1(S_2, \delta) + \left|\widehat{E}(h', S_1') - \widehat{E}(h', S_2')\right|. \end{aligned}$$

In particular, with probability $\geq 1 - 2\delta$,

$$\begin{aligned} &P\left[\left|\widehat{E}(h, S_1) - \widehat{E}(h, S_2)\right| > \varepsilon\right] \\ &\quad\leq P\left[\varepsilon_1(S_1, \delta) + \varepsilon_1(S_2, \delta) + \left|\widehat{E}(h', S_1') - \widehat{E}(h', S_2')\right| > \varepsilon\right] \\ &\quad= P\left[\left|\widehat{E}(h', S_1') - \widehat{E}(h', S_2')\right| > \varepsilon - \varepsilon_1(S_1, \delta) - \varepsilon_1(S_2, \delta)\right]. \end{aligned}$$

We have obtained a bound on the additional classification error that is incurred when projecting the sample down from some n dimensional space to a k dimensional space. As a result, we have established the fact that the difference between the classification performance on two samples, in high dimension, is very similar to the difference in low dimension. This is now used to prove the main result of the chapter.

Proof. (**Theorem 3.4**) Let S_1 denote a sample of size m from $\Re^n$. Let $\mathcal{H}$ denotes the space of all linear classifiers in n dimensional space and let $h \in \mathcal{H}$. Also, let $\overline{E}(h)$ denote the expected error of a classifier h, on the data sampled according to distribution $\mathcal{D}$ and $\widehat{E}(h, S_1)$, the empirical (observed) error of the same classifier, h, on the sample set S_1 when the data was sampled according to the same distribution $\mathcal{D}$.

To obtain the generalization error of a classifier which is learned in a high dimensional space, we want to compute the bound on the following quantity as a function of the error ε:

$$P\left[\sup_{h \in \mathcal{H}} \left|\overline{E}(h) - \widehat{E}(h, S_1)\right| > \varepsilon\right]. \tag{3.7}$$

We are interested in the probability that uniformly all classifiers from the hypothesis space $\mathcal{H}$ will do well on future data. To compute the bound, we use the technique of double samples which has been used in proving bounds of this kind. Assume we observe a sample of size $2m$, where S_1, S_2 denote the first and second half of sample resp. From [Vapnik, 1998, 131–133]:

$$P\left[\sup_{h\in\mathcal{H}}\left|\overline{E}(h)–\widehat{E}(h,S_1)\right|>\varepsilon\right]\le 2P\left[\sup_{h\in\mathcal{H}}\left|\widehat{E}(h,S_1)–\widehat{E}(h,S_2)\right|>\frac{\varepsilon}{2}\right].$$

Now suppose we project the data along with the hyperplane down to k dimensional space, using a random projection matrix. Then according to Lemma 3.8, with high probability most of the data stays on the correct side of the hyperplane. Formally, with probability $\geq 1-2\delta$,

$$P\left[\sup_{h\in\mathcal{H}}\left|\widehat{E}(h,S_1)–\widehat{E}(h,S_2)\right|>\frac{\varepsilon}{2}\right]\le P\left[\sup_{h\in\mathcal{H}}\left|\widehat{E}(h',S_1')–\widehat{E}(h',S_2')\right|>\rho\right],$$

where $\rho = \frac{\varepsilon}{2} - \varepsilon_1(S_1,\delta) - \varepsilon_1(S_2,\delta)$.

Since the sample sets S_1, S_2 contains independent samples so do S_1', S_2', and using results from [Vapnik, 1998, p. 134] we write

$$\begin{aligned}P\left[\sup_{h\in\mathcal{H}}\left|\widehat{E}(h',S_1')–\widehat{E}(h',S_2')\right|>\rho\right] &= P\left[\sup_{h'\in\mathcal{H}'}\left|\widehat{E}(h',S_1')–\widehat{E}(h',S_2')\right|>\rho\right]\\ &\le \sum_{h'\in\mathcal{H}'} P\left[\left|\widehat{E}(h',S_1')–\widehat{E}(h',S_2')\right|>\rho\right]\\ &\le N^k(2m)\exp(-2\rho^2 m)\\ &\le \left(\frac{2em}{k+1}\right)^{k+1}\exp(-2\rho^2 m),\end{aligned}$$

where $N^k(2m)$ is the maximum number of different partitions of $2m$ samples of k dimensional data that can be realized by a k dimensional linear classifier. The last inequality uses the Sauer's lemma to bound this quantity as a function of the VC dimension of a classifier which in this case happens to be $(k+1)$. To bound this probability by δ, the confidence parameter, we isolate ρ from the inequality, which gives $\left(\frac{2em}{k+1}\right)^{k+1}\exp(-2\rho^2 m) \le \delta$. Simplification reveals that this inequality holds for

$$\rho = \sqrt{\frac{(k+1)\ln\left(\frac{2em}{k+1}\right)+\ln\left(\frac{1}{\delta}\right)}{2m}}.$$

Solving for ε, we have $\varepsilon = 2\rho + 2\varepsilon_1(S_1,\delta) + 2\varepsilon_1(S_2,\delta)$. Putting it together, we get

$$P\left[\sup_{h\in\mathcal{H}}\left|\overline{E}(h) - \widehat{E}(h,S_1)\right| > \varepsilon\right] \le 2\delta.$$

Combining this, together with the bound on the confidence that Eq. (3.7) holds, we get that with probability $\geq (1-2\delta)(1-2\delta) \geq 1-4\delta$, we have

$$\begin{aligned}\overline{E}(h,S_1) &\leq \widehat{E}(h,S_1)+\varepsilon \\ &\leq \widehat{E}(h,S_1)+2\varepsilon_1(S_1,\delta)+2\varepsilon_1(S_2,\delta) \\ &\quad +2\sqrt{\frac{(k+1)\ln\left(\frac{2em}{k+1}\right)+\ln\left(\frac{1}{\delta}\right)}{2m}}.\end{aligned}$$

Plugging in the value of ρ along with the value of $\varepsilon_1(\cdot,\cdot)$, gives the final bound.

In fact, it is possible to improve the bound for some ranges of k, using the fact that we only care about the distortion in the distance of points from the classifier, rather than the distortion in the size of the projected vectors themselves, as in Lemma 3.5. This is formalized next.

LEMMA 3.9 *Let $x' = Rx$ and $h' = Rh$. Let $\nu = h^T x$ and $\nu' = h'^T x'$, where R is $n \times k$ random projection matrix. Then, we have*

$$E[\nu'] = \nu \quad \textit{and} \quad \mathrm{var}[\nu'] = \frac{1+\nu^2}{k}. \tag{3.8}$$

Proof. Let $R = \{r_1, r_2, ..., r_k\}$, where r_i are the row vectors (see Definition 3.2). Then, $\nu' = h'^T x' = \sum_{i=1}^k h^T r_i r_i^T x$. Let $\nu_i' = h^T r_i r_i^T x$. Clearly, the ν_i's are independent random variables, and we can express $\nu' = \sum_{i=1}^k \nu'_i$ as the sum of independent random variables. Furthermore,

$$E[\nu_i'] = E[h^T r_i r_i^T x] = h^T E[r_i r_i^T] x = h^T (I/k) x = h^T x = \nu/k,$$

where I is the identity matrix. This holds as each entry of the random projection matrix is $\sim N(0,1/k)$, and thus the matrix $E[r_i r_i^T]$ has zero in each non-diagonal entry, as it is the expectation of the product of two independent variables, each of expectation zero, and the value of a diagonal entry is the second moment of $N(0,1/k)$, which is $1/k$. In particular, $E[\nu'] = E[\sum_i \nu_i'] = \nu$.

We next compute $\mathrm{var}[\nu'_i]$. Let $r_i = [\mu_1, \ldots, \mu_n]$, where $\mu_j \sim N(0,1/k)$. Then:

$$\begin{aligned}E[\nu_i'^2] &= E[(h^T r_i r_i^T x)^2] \\ &= E[(h^T r_i r_i^T x)(h^T r_i r_i^T x)] \\ &= \sum_{a=1}^n \sum_{b=1}^n \sum_{c=1}^n \sum_{d=1}^n x_a x_b h_c h_d E[\mu_a \mu_b \mu_c \mu_d].\end{aligned}$$

Note, that if a is not equal to either b, c, or d, then $E[\mu_a \mu_b \mu_c \mu_d] = E[\mu_a] E[\mu_b \mu_c \mu_d] = 0 \cdot E[\mu_b \mu_c \mu_d]$, as μ_a is independent of μ_b, μ_c, μ_d. We can apply the same argumentation to b, c, and d. It follows, thus, that the only non-zero terms in the above summation are when: (i) $a = b$ and $c = d$, or (ii) $a = c$ and $b = d$, or (iii) $a = d$ and $b = c$ (note that the case $a = b = c = d$ is counted three times in those cases). Thus

$$\begin{aligned} E\Big[\nu_i'^2\Big] &= \sum_{a=1}^{n}\sum_{c=1}^{n} x_a x_a h_c h_c E[\mu_a\mu_a\mu_c\mu_c] + \sum_{a=1}^{n}\sum_{b=1}^{n} x_a x_b h_a h_b E[\mu_a\mu_b\mu_a\mu_b] \\ &\quad + \sum_{a=1}^{n}\sum_{b=1}^{n} x_a x_b h_b h_a E[\mu_a\mu_b\mu_b\mu_a] - 2\sum_{a=1}^{n} x_a x_a h_a h_a E[\mu_a\mu_a\mu_a\mu_a]. \end{aligned}$$

We observe that $E[\mu_u^2] = 1/k$ and $E[\mu_u^4] = 3/k^2$. Thus, $E[\mu_a^2\mu_b^2] = 1/k^2$ if $a \neq b$ and $E[\mu_a^2\mu_b^2] = 3/k^2$ if $a = b$. Thus,

$$\begin{aligned} E\Big[\nu_i'^2\Big] &\leq \frac{1}{k^2}\sum_{a=1}^{n}\sum_{c=1}^{n} x_a x_a h_c h_c + \frac{1}{k^2}\sum_{a=1}^{n}\sum_{b=1}^{n} x_a x_b h_a h_b \\ &\quad + \frac{1}{k^2}\sum_{a=1}^{n}\sum_{b=1}^{n} x_a x_b h_b h_a + 3\frac{2}{k^2}\sum_{a=1}^{n} x_a x_a h_a h_a - 2\frac{3}{k^2}\sum_{a=1}^{n} x_a x_a h_a h_a \\ &= \frac{1}{k^2}\left(||x||^2||h||^2 + 2||xh||^2\right) = \frac{1+2\nu^2}{k^2}, \end{aligned}$$

as $||x|| = ||h|| = 1$. Finally,

$$\begin{aligned} \mathrm{var}\Big[\nu_i'\Big] &= E\Big[\nu_i'^2\Big] - \Big(E\Big[\nu_i'\Big]\Big)^2 = \frac{1+2\nu^2}{k^2} - \frac{\nu^2}{k^2} \\ &= \frac{1+\nu^2}{k^2}. \end{aligned}$$

We conclude that $\mathrm{var}[\nu'] = k \cdot \mathrm{var}[\nu_i'] = \frac{1+\nu^2}{k}$.

Using Chebyshev bound we get:

LEMMA 3.10 *Let R be a random projection matrix as in Definition 3.2, $x' = Rx, h' = Rh, \nu = h^T x, \nu' = h'^T x'$. Then,*

$$P\Big[sign(\nu) \neq sign(\nu')\Big] \leq \frac{2}{k\nu^2}.$$

Proof. Using the Chebyshev bound, we know that

$$P\left[|\nu' - E[\nu']| \geq \frac{\varepsilon}{\sigma(\nu')}\sigma(\nu')\right] \leq \frac{(\sigma(\nu'))^2}{\varepsilon^2}, \tag{3.9}$$

where $\sigma(\nu')$ is the standard deviation of ν'. Plugging in the bounds of Lemma 3.9, we have

$$P\Big[|\nu' - \nu| \geq \varepsilon\Big] \leq \frac{2}{k\varepsilon^2}. \tag{3.10}$$

Now, the $sign(\nu) \neq sign(\nu')$ only if $|\nu' - \nu| \geq |\nu|$. Which implies that

$$P\Big[sign(\nu) \neq sign(\nu')\Big] \leq P(|\nu' - \nu| > |\nu|) \leq \frac{2}{k\nu^2}. \tag{3.11}$$

Note that the difference between this and the result in Lemma 3.5 is that there we used the Chernoff bound, which is tighter for large values of k. For smaller values of k the above result will provide a better bound.

This result can be further improved if the random projection matrix used has entries in $\{-1, +1\}$, using a variation of Lemma 3.9 along with a recent result [Achlioptas, 2001].

4. Analysis

Based on Lemma 3.5 and Lemma 3.10, the expected probability of error for a k-dimensional image of x, given that the point x is at distance $\nu(x)$ from the n-dimensional hyperplane (where the expectation is with respect to selecting a random projection) is given by

$$\min\left(3\exp\left(-\frac{(\nu(x))^2 k}{8(2 + |(\nu(x))|)^2}\right), \frac{2}{kl^2}, 1\right). \tag{3.12}$$

This expression measures the contribution of the point to the generalization error, as a function of its distance from the hyperplane (and for a fixed k). Figure 3.1 shows this term (Eqn 3.12) for different values of k. This bell shaped curve, depicting the results, exhibits that all points have some contribution to the error, and the relative contribution of a point decays exponentially fast as a function of its distance from the hyperplane. Given the probability distribution over the instance space and a fixed classifier, one can compute the distribution over the margin which is then used to compute the projection profile of the data as

$$\int_{x \in \mathcal{D}} \min\left(3\exp\left(-\frac{(\nu(x))^2 k}{8(2 + |(\nu(x))|)^2}\right), \frac{2}{kl^2}, 1\right) d\mathcal{D}. \tag{3.13}$$

Consider, for example, the case when the distribution over the distance of the points from the hyperplane, D_l is normal with mean μ_l and variance σ_l (since the distance is bounded in $[0, 1]$, for the analysis we need to make sure that means and variances are such that no points lies outside this region). In this case, the projection profile can be computed analytically. Figure 3.2 shows the bound for this case (mean = 0.3, variance = 0.1) as a function of k (the projected dimension of the data) and compares the VC-dimension term with the random projection term. It is evident that when the dimension of the data is very small, it is better to consider the VC-dimension based bounds but as soon as the dimension of the data increases, the VC-dimension term is much larger. Our bound can be thought of as doing a tradeoff between the two terms.

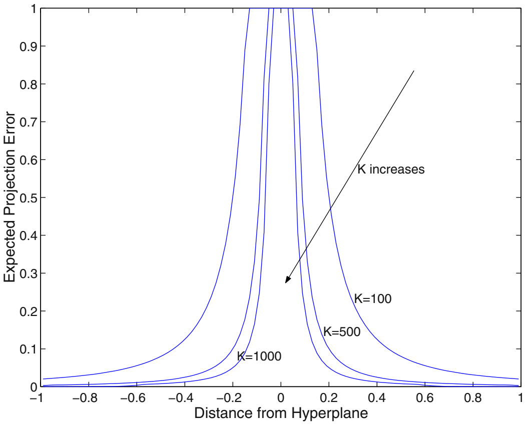


Figure 3.1. The contribution of data points to the generalization error as a function of their distance from the hyperplane.

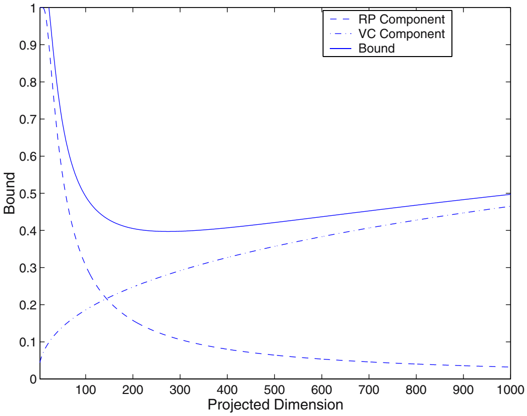


Figure 3.2. The bound obtained when the margin has a normal distribution. It shows the tradeoff between the VC-dimension and the random projection terms in the bound

4.1 Comparison with Existing Bounds

The most basic generalization bound is the one derived from VC-theory [Blumer et al., 1989]. The true error of an n-dimensional linear classifier whose empirical error on a sample of size m is $\widehat{\varepsilon}$ is bounded, with probability at least $1 - \delta$ by,

$$\varepsilon \leq \widehat{\varepsilon} + \sqrt{\frac{(n+1)(\ln(\frac{2m}{n+1}) + 1) - \ln\frac{\delta}{4}}{m}}, \tag{3.14}$$

where we use the fact that the VC-dimension of a linear classifier is $n+1$. This bound is very general and gives the worst case generalization performance of the classifier. It depends only on the number of samples and the dimensionality of the data.

Shawe-Taylor [Shawe-Taylor, 1998] and Shawe-Taylor and Cristianini [Shawe-Taylor and Cristianini, 1999] have explored a new direction in which the margin of the data is used to derive bounds on the generalization error. Their bound depends on the fat-shattering function, *afat*, a generalization of the VC dimension, introduced in [Kearns and Schapire, 1994]. For sample size m the bound is given by:

$$\varepsilon \leq \frac{2}{m}\left(f \log_2(32m) \log_2 \frac{8em}{f} + \log_2 \frac{8m}{\delta}\right), \tag{3.15}$$

where $f = \mathit{afat}(\delta/8)$ and δ is the minimum margin of data points in the sample. For linear functions this is bounded by $(BR/\delta)^2$, where B is the norm of the classifier and R is the maximal norm of the data.

It is useful to observe the key difference between these two bounds; while the former depends only on the space in which the data lies and is totally independent of the actual data, the latter is independent of this space and depends on the performance (margin) of the classifier on the given data.

The new bound proposed here can be thought of as providing a link between the two existing bounds described above. The first component is a function of the data and independent of the true dimension whereas the second component is a function of the projected dimension of the data.

In most cases, the bounds in Eqs. 3.14 and 3.15 are weak in the sense that the amount of data required before the bound is meaningful (< 0.5) is huge. For the VC-dimension based bounds, for example, the amount of data required for a meaningful bound is at least 17 times the dimension of the data. This is not feasible in many natural language processing and computer vision tasks where data dimensionality may be very high [Roth, 1998]. Figure 3.3 provides some quantitative assessment of these claims. It gives the number of samples required before the fat-shattering bound is meaningful (< 0.5). This shows that even for the case when the margin is 0.9, almost a hundred thousand points need to be observed before the bounds are useful.

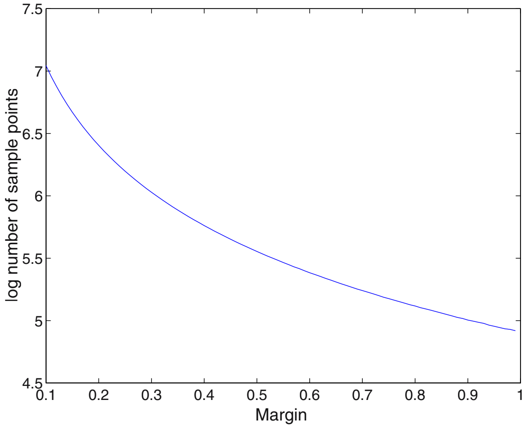


Figure 3.3. The number of sample points required before the fat-shattering bound in Eqn 3.15 is meaningful, as a function of margin. Here the logarithm is in base 10.

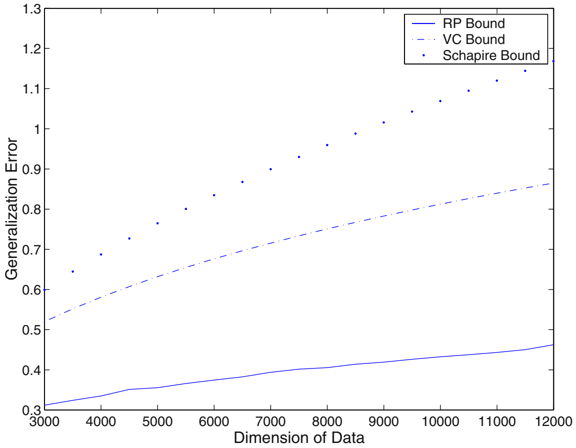


Figure 3.4. The generalization bound for the (10-100-N) threshold function.

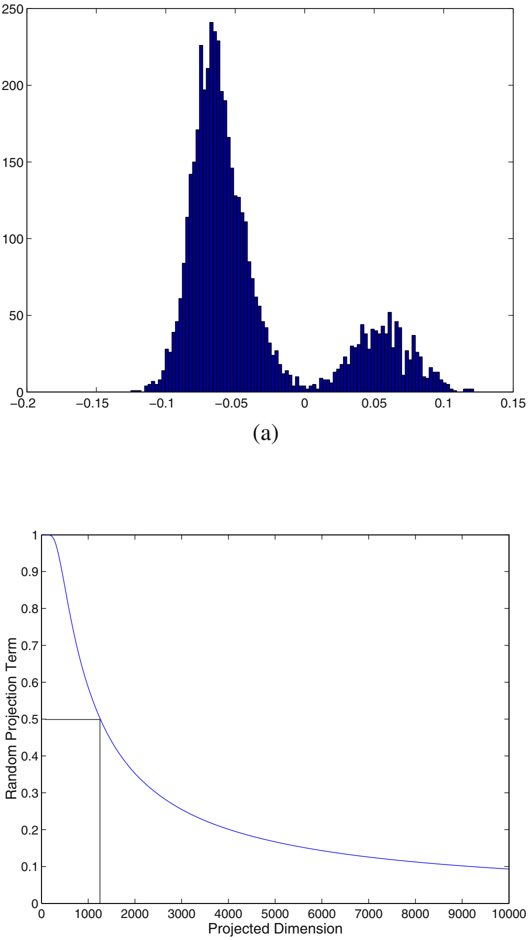
Schapire et al. [Schapire et al., 1997] have developed bounds explaining the good performance of the voting methods. They have argued that the good performance of boosting based classifiers can be attributed to large margin of the learned classifier and have derived bounds based on the margin of the data. Their bound is

$$\varepsilon \leq P_S[\nu \leq \theta] + O\left(\frac{1}{\sqrt{m}}\left(\frac{d\log^2\left(\frac{m}{d}\right)}{\theta^2} + \log\left(\frac{1}{\delta}\right)\right)^{1/2}\right). \qquad (3.16)$$

There is a striking similarity between our bound and this. The first term in their bound assumes that we make error on all points which have margin less than θ, whereas in our case, the projection profile defines the amount of error that is made. The second term of their bound gives the generalization error when the classification is done with a good margin.

We compared this bound to the random projection bound on simulated data. We generated the data according to the concept $(r-m-n)$ threshold function. The results are shown in Figure 3.4. In this experiment, $50,000$ training points are assumed. The figure also shows the VC-bound for this data.

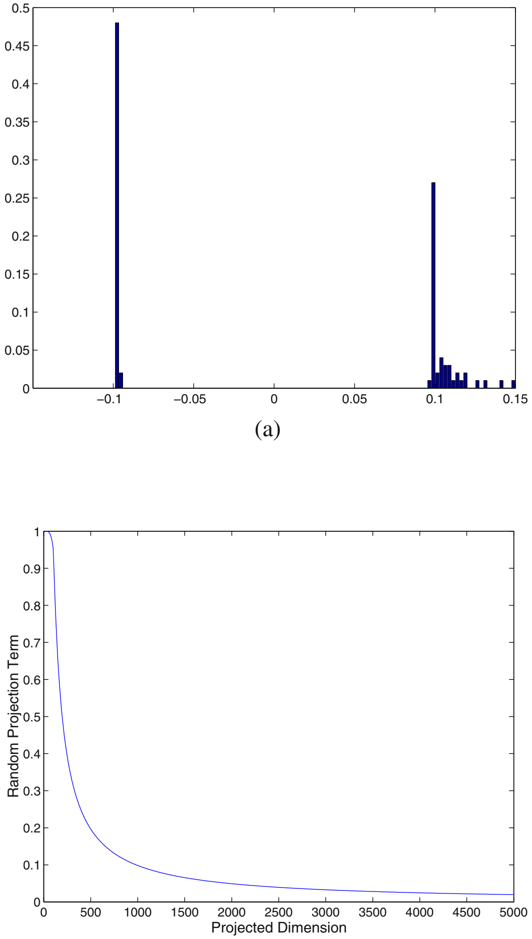
We also compared the performance of the new bound with respect to the existing bounds on some real problems. In the first experiment, we considered 17000 dimensional data taken from the problem of context sensitive spelling correction [Golding and Roth, 1999]. A winnow based algorithm, which was shown very successful on this problem, was used to learn a linear classifier. Figure 3.5 shows the histogram of the distance of the data with respect to the learned classifier. It is evident that a large number of data points are very close to the classifier and therefore the fat-shattering bounds are not useful. At the same time, to gain confidence from the VC-dimension based bounds, we need over $120,000$ data points. Figure 3.5 shows the random projection term for this case; this term is below 0.5 already after 2000 samples and thus for the overall bound to be small, we need much less examples as compared to the VC-dimension case. The second experiment, considered the problem of face detection. A Radial Basis Function (RBF) kernel was used to learn the classifier. Figure 3.6(a) shows the histogram of the margin of the learned classifier and Figure 3.6(b) gives the random projection term as a function of the dimension of the data.



(a)

(b)

Figure 3.5. (a) Histogram of the distance of the points from the classifier for the context sensitive spelling correction. (b) The distortion error due to random projection as a function of the dimension of the projected space.



(a)

(b)

Figure 3.6. (a) Histogram of the distance of the points from the classifier for the face detection experiment. (b) The distortion error due to random projection as a function of the dimension of the projected space.

5. Summary

We have presented a new analysis method for linear learning algorithms that uses random projections and margin distribution analysis. The main contribution of the chapter are:

1 using this method to develop a new data dependent complexity measure for learning, and

2 introducing a bound on the true error of a learning algorithm, as a function of the margin distribution of the data relative to the learned classifier.

While the random projection method was used in this chapter as an analysis tool, one of the main direction of future research is to investigate algorithmic implications of the ideas presented here. In addition, we plan to study the bounds on real data sets and develop a better understanding of the projection profile introduced here.

Chapter 4

THEORY: SEMI-SUPERVISED LEARNING

Many pattern recognition and human computer interaction applications require the design of classifiers. classifiers are either designed from expert knowledge or from training data which can be either labeled or unlabeled. In many applications, obtaining fully labeled training sets is a difficult task since labeling is done using human expertise, which is expensive, time consuming, and error prone. Obtaining unlabeled data is usually easier since it involves the collection of data that are known to belong to one of the classes without having to label it.

In this chapter, we discuss training probabilistic classifiers with labeled and unlabeled data. This method of learning is known as semi-supervised learning. We provide a new analysis that shows under what conditions unlabeled data can be used in learning to improve classification performance. We also show that if the conditions are violated, using unlabeled data can be detrimental to classification performance.

In Chapter 7, we will discuss the implications of this analysis to a specific type of probabilistic classifiers, Bayesian networks, and we will present a new structure learning algorithm that can utilize unlabeled data to improve classification. Later, in Chapters 10 and 11 we show how the resulting algorithms are successfully employed in two applications related to human computer interaction and pattern recognition: facial expression recognition and face detection.

1. Introduction

Is there value to unlabeled data in supervised learning of classifiers? This fundamental question has been increasingly discussed in recent years, with a general optimistic view that unlabeled data hold great value. Citing an increasing number of applications and algorithms successfully utilizing unlabeled data [Blum and Mitchell, 1998; Nigam et al., 2000; Shahshahani and Landgrebe, 1994a; Baluja, 1998; Bruce, 2001; Bennett and Demiriz, 1998; Ghani, 2002; Seeger, 2001] further magnified by theoretical content proving of the value of unlabeled data in certain cases [Castelli, 1994; Ratsaby and Venkatesh, 1995; Cooper and Freeman, 1970; O'Neill, 1978], semi-supervised learning is seen optimistically as a learning paradigm that can relieve the practitioner from the need to collect many expensive labeled training data.

However, several disparate empirical evidence in the literature suggest that there are situations in which the addition of unlabeled data to a pool of labeled data, causes degradation of the classifier's performance [Nigam et al., 2000; Shahshahani and Landgrebe, 1994a; Baluja, 1998; Bruce, 2001], in contrast to improvement of performance when adding additional labeled data. In this chapter, we provide an analysis of semi-supervised learning using maximum likelihood estimators which explains all the results seen in the literature: when unlabeled data improves and degrades performance.

Our analysis is based on classifiers represented by a distribution, who's parameters are estimated from data and plugged into the optimal decision rule (Section 4.2). The assumed parametric family of the classifier leads to the definition of correct and incorrect models. The existing literature (Section 4.3) is generally optimistic regarding the positive use of unlabeled data, however, existing theoretical results always assumed that the model is correct. The question is what happens when the model is not correct. We answer this question by providing the unified asymptotic properties of maximum likelihood (ML) estimation (Section 4.5). When the model is correct, the asymptotic properties of ML coincide with existing theoretical results, showing that unlabeled data can always be used profitably to learn a classifier. When the model is incorrect, the properties suggest that ML estimators suffer from an odd lack of robustness with respect to the ratio between labeled and unlabeled data. As the ratio between labeled and unlabeled data changes, so do the asymptotic estimates of the classifier's parameters and the estimation bias from the true data generating distribution. We show, using examples, that the increase in estimation bias leads to an increase in classification error with unlabeled data. The examples prove that unlabeled data can degrade performance with incorrect models, even if numerical problems are removed.

With finite training data the situation becomes slightly more complex. Experiments with artificial and real data sets show that unlabeled data will usually degrade performance when incorrect models are assumed, and will improve performance with correct modeling assumptions (Section 4.6). However, there are cases with incorrect models where unlabeled data are observed to improve the classification performance. We show experiments of such cases, arguing that the most probable cause is the fact that the decrease in variance when adding unlabeled data has a greater effect than the increase in bias, especially when there are a large number of parameters to estimate and a very small number of labeled data. This explanation could explain some of the existing successes in the literature.

Leveraging on the properties of semi-supervised learning with incorrect models, we suggest that unlabeled data can be used for validating modeling assumptions (Section 4.6.4); observing that unlabeled data degrade a classifier's performance is a strong indication that the assumed model is incorrect.

2. Properties of Classification

We briefly repeat (with some change of notation) the definitions and properties of classification given previously in Chapter 2. The goal is to classify an incoming vector of observables $\mathbf{X}$. Each instantiation of $\mathbf{X}$ is a *sample*. There exists a *class variable* C; the values of C are the *classes*. Let $P(C, \mathbf{X})$ be the *true* joint distribution of the class and features from which any a sample of some or all of the variables from the set $\{C, X\}$ is drawn, and $p(C, \mathbf{X})$ be the density distribution associated with it. We want to build *classifiers* that receive a sample $\mathbf{x}$ and output either one of the values of C.

We assume 0-1 loss and consequently, the objective is to minimize the probability of classification error. The optimal classification rule, under the 0-1 cost function, is the maximum a-posteriori (MAP) rule [Devroye et al., 1996]:

$$g^*(\mathbf{x}) = max_{c'} \left[p(C = c'|\mathbf{x}) \right], \tag{4.1}$$

where c' goes over all values of C and $p(C|\mathbf{X})$ is the a-posteriori distribution of C which could be computed from the joint distribution using

$$p(C = c'|\mathbf{x}) = \frac{p(C = c, \mathbf{X} = x)}{p(\mathbf{X} = x)}.$$

An error is made when $g^*(\mathbf{x}) \neq c$, where c is the true label of the sample $\mathbf{x}$. The maximum a-posteriori (MAP) decision rule is optimal in the sense that for every tuple $(c, \mathbf{x})$ and any other classifier $g(x)$ which maps $\mathbf{x}$ to the class labels the following holds [Devroye et al., 1996]:

$$P(g^*(\mathbf{x}) \neq c) \leq P(g(\mathbf{x}) \neq c). \tag{4.2}$$

The decision rule $g^*(\mathbf{x})$ is called the Bayes rule and $\mathbf{e}_{\mathrm{B}} = P(g^*(\mathbf{X}) \neq C)$ is known as the Bayes error (or Bayes risk), and is the minimum probability of error achievable for the given classification problem. No other classification rule can achieve a lower error.

The Bayes error is achievable when the a-posteriori probability $p(C|\mathbf{X})$ is available (or $g^*(\mathbf{x})$). In reality, neither $p(C|\mathbf{X})$ nor $g^*(x)$ are known, thus, we try to estimate $p(C|\mathbf{X})$ from data (of size N). We assume a parametric model $P(C, \mathbf{X}|\theta)$, with parameters θ defined in a compact subset of Euclidean space denoted by Θ. An estimate of θ is denoted by $\hat{\theta}_N$. The process of estimating

$\hat{\theta}_N$ is the learning rule. Given $p\Big(C|\mathbf{X},\hat{\theta}_N\Big)$, the probability density induced by the parametric model, the estimates are plugged into the optimal classification rule:

$$g_{\hat{\theta}_N}(\mathbf{x}) = \max_{c'}\left[p\Big(C = c'|\mathbf{x},\hat{\theta}_N\Big)\right], \tag{4.3}$$

with an associated probability of error given by $\mathbf{e}(\hat{\theta}) = P\Big(g_{\hat{\theta}_N}(\mathbf{X}) \neq C\Big)$. Clearly, $\mathbf{e}_{\text{B}} \leq \mathbf{e}(\hat{\theta}_N)$ for any size of training data N. A consistent learning rule is one that yields $lim_{N\rightarrow\infty}E[\mathbf{e}(\hat{\theta}_N)] = \mathbf{e}_{\text{B}}$, and thus can achieve the Bayes error as the size of the training set increases.

If the distribution $P(C,\mathbf{X})$ belongs to the family $P(C,\mathbf{X}|\theta)$, we say the "model is correct"; otherwise we say the "model is incorrect."

When the model is correct, the difference between the expected value $E_\theta\left[\hat{\theta}\right]$ and the true parameter set, denoted as $\theta_\top$, is called the *estimation bias*. When the estimation bias is zero, the estimator $\hat{\theta}_N$ is *unbiased*, and when the estimation bias goes to 0 as N approaches infinity, the estimator is consistent. In both cases, the learning rule is also consistent (i.e., the estimated classifier achieves the Bayes error rate).

When the model is incorrect, we denote bias loosely to be the the distance between $P(C,\mathbf{X})$ and the estimated $P\Big(C,\mathbf{X}|\hat{\theta}\Big)$. Different distance measures can be used, such as mean square error, Kullback-Leiber (KL) divergence and others. In the framework of maximum likelihood estimators, the relevant distance is the KL divergence.

The difference between the expected value of $\mathbf{e}(\hat{\theta})$ and $\mathbf{e}_{\text{B}}$ is called *classification bias*. When the model is correct and for a consistent learning rule, the bias is zero. When the model is incorrect, there exists a θ for which the classification bias is minimized. The bias is generally not zero, although it could be zero for some cases. We discuss the relationship between classification bias and estimation bias in the following sections.

3. Existing Literature

The existing literature presents several empirical and theoretical findings that indicate the positive value of unlabeled data. Cooper and Freeman were optimistic enough about unlabeled data so as to title their work as "On the asymptotic improvement in the outcome of supervised learning provided by additional nonsupervised learning" [Cooper and Freeman, 1970]. Other early studies, such as [Hosmer, 1973; O'Neill, 1978] further strengthened the assertion that unlabeled data should be used whenever available.

Castelli and Cover [Castelli, 1994; Castelli and Cover, 1995; Castelli and Cover, 1996] and Ratsaby and Venkatesh [Ratsaby and Venkatesh, 1995] consider situations where unlabeled samples are used to estimate the decision regions (by estimating the form of $p(C, \mathbf{X})$) and labeled samples are used solely to determine the labels of each decision region. This procedure is called "Algorithm M" by Ratsaby and Venkatesh; they use probably approximately correct (PAC)-learning to prove that Algorithm M requires exponentially more unlabeled samples than labeled samples to obtain a given classification performance for models in the exponential family. Castelli and Cover work directly from optimal Bayesian procedures and do not take the detour of "plug-in" estimation procedures. They prove that Algorithm M is asymptotically optimal under various assumptions. The main message of their work is that, asymptotically, labeled data contribute exponentially faster than unlabeled data to the reduction of classification error. Krishnan and Nandy [Krishnan and Nandy, 1990a; Krishnan and Nandy, 1990b] extened the results of [Ganesalingam and McLachlan, 1978] to provide efficiency results for discriminant and logistic-normal models for samples that are labeled stochastically.

It should be noted that such previous theoretical work makes the critical assumption that $p(C, \mathbf{X})$ belongs to the family of models $p(C, \mathbf{X}|\theta)$ (that is, the "model is correct").

There has been plenty of recent applied work with semi-supervised learning. Shahshahani and Landgrebe describe classification improvements with unlabeled spectral data [Shahshahani and Landgrebe, 1994a]; Mitchell and co-workers report a number of approaches to extract valuable information from unlabeled data, from variations of maximum likelihood estimation [Nigam et al., 2000] to co-training algorithms [Mitchell, 1999]. Other publications report on EM-like algorithms [Baluja, 1998; Bruce, 2001; Miller and Uyar, 1996] and co-training approaches [Collins and Singer, 2000; Comite et al., 1999; Goldman and Zhou, 2000]. There have also been workshops on the labeled-unlabeled data problem (at NIPS1999, NIPS2000, NIPS2001, and IJCAI2001). Overall, these publications and meetings advance an optimistic view of the labeled-unlabeled data problem, where unlabeled data can be profitably used whenever available.

However, a more detailed analysis of current empirical results does reveal some puzzling aspects of unlabeled data. Four results are particularly interesting:

1 Shahshahani and Landgrebe [Shahshahani and Landgrebe, 1994a] report experiments where unlabeled data degraded performance. They attribute such cases to deviations from modeling assumptions, such as outliers and "samples of unknown classes" — they even suggest that unlabeled samples should be used with care, and only when the labeled data alone produce a poor classifier.

2 Baluja [Baluja, 1998] used Naive Bayes [Friedman, 1997] and Tree-Augmented Naive Bayes (TAN) classifiers [Friedman et al., 1997] to obtain

excellent classification results, but there were cases where unlabeled data degraded performance.

3 In work aimed at classification of documents, Nigam et al [Nigam et al., 2000] used Naive Bayes classifiers and a large number of observables. They discussed several situations where unlabeled data degraded performance, and proposed techniques to reduce the observed degradation.

4 Bruce [Bruce, 2001] used labeled and unlabeled data to learn Bayesian network classifiers, from Naive Bayes classifiers to fully connected networks. The Naive Bayes classifier displayed bad classification performance, and in fact the performance degraded as more unlabeled data were used. In some cases, more complex networks also displayed degradation as unlabeled samples were added.

Both Shahshahani and Landgrebe [Shahshahani and Landgrebe, 1994b] and Nigam [Nigam, 2001] are rather explicit in stating that unlabeled data can degrade performance, but rather vague in explaining how this can be so and how to analyze the phenomenon. They refer to numerical problems and modeling errors as potential sources of difficulties.

The rest of the chapter concentrates on the more fundamental problem of incorrect modeling assumptions. We do not deny that numerical problems can happen in practice; interesting discussions of numerical problems in semi-supervised learning can be found elsewhere ([McLachlan and Basford, 1988] and [Corduneanu and Jaakkola, 2002]). The theory in Section 4.5 allows us to analyze semi-supervised learning without resorting to numerical methods, and thus obtain insights that are not clouded by the uncertainties of numerical optimization. The examples in Section 4.5.3 show that performance degradation with unlabeled data would occur even if numerical problems were somehow removed.

4. Semi-supervised Learning Using Maximum Likelihood Estimation

In semi-supervised learning, classifiers are built from a combination of N_l labeled and N_u unlabeled samples. We assume that the samples are independent and identically distributed.

We consider the following scenario. A sample $(c, \mathbf{x})$ is generated from $p(C, \mathbf{X})$. The value c is then either known, and the sample is a *labeled* one; or the value c is hidden, and the sample is an *unlabeled* one. The probability that any sample is labeled, denoted by λ, is fixed, known, and independent of the samples[1]. Thus, the same underlying distribution $p(C, \mathbf{X})$ models both labeled and unlabeled data; we do not consider the possibility that labeled and unlabeled samples have different generating mechanisms. It is worth noting that we assume that the revealed label is correct and it is not corrupted by noise; the case of noisy labels has been studied in various works (such as [Chhikara and McKeon, 1984; Chittineni, 1981; Krishnan and Nandy, 1990a; Krishnan and Nandy, 1990b], Chapter 2 of [Pal and Pal, 2002]).

The likelihood of a labeled sample $(c, \mathbf{x})$ is equal to $\lambda p(c, \mathbf{x}|\theta)$. An unlabeled sample $\mathbf{x}$ does not contain information about the class variable, and its likelihood is $(1-\lambda)p(\mathbf{x}|\theta)$. Here $p(\mathbf{X}|\theta)$ is the marginal for $\mathbf{X}$, a mixture model:

$$p(\mathbf{X}|\theta) = \sum_C p(C, \mathbf{X}|\theta) = \sum_C p(\mathbf{X}|C, \theta)\, p(C|\theta) \tag{4.4}$$

We have the following assumption:

ASSUMPTION 4.1 *The mixtures expressed by Eqn. (4.4) are identifiable [Redner and Walker, 1984]; that is, distinct parameter values determine distinct members of the family, with the understanding that permutations of the mixtures components are allowed.*

The distribution $p(C, \mathbf{X}|\theta)$ can be decomposed either as $p(C|\mathbf{X}, \theta)\, p(\mathbf{X}|\theta)$ or as $p(\mathbf{X}|C, \theta)\, p(C|\theta)$, which lead to two paradigms for choosing the parametric model: *generative* and *diagnostic*.

A generative model is a parametric model where both $p(\mathbf{X}|C, \theta)$ and $p(C|\theta)$ depend explicitly on θ[2]. The log-likelihood function of a generative model for a dataset with labeled and unlabeled data is:

$$L(\theta) \quad = \quad L_l(\theta) + L_u(\theta) + \log\left(\lambda^{N_l}(1-\lambda)^{N_u}\right), \tag{4.5}$$

where

$$\begin{aligned} L_l(\theta) &= \sum_{i=1}^{N_l} \log\left[\prod_C p\big(c'|\theta\big)\, p\big(\mathbf{x}_i|c', \theta\big)\big)^{I_{\{C=c'\}}(c_i)}\right] \\ L_u(\theta) &= \sum_{j=(N_l+1)}^{N_l+N_u} \log\left[\sum_C p\big(c'|\theta\big)\, p\big(\mathbf{x}_j|c', \theta\big)\right], \end{aligned}$$

and $I_A(Z)$ is the indicator function (1 if $Z \in A$; 0 otherwise), $\sum_C$ and $\prod_C$ are computed over all values of C, and for convenience we assume the data is ordered so that the first N_l samples are labeled.

[1] This is different from [Nigam et al., 2000] and [Corduneanu and Jaakkola, 2002], where λ is a parameter that can be set.

[2] The term *sampling model* [Dawid, 1976] and *type I* [Zhang and Oles, 2000] have been used to refer to generative models.

Diagnostic models focus only on $p(C|\mathbf{X},\theta)$ and take the marginal $p(\mathbf{X})$ to be independent of θ. For example, logistic regression estimates $p(C|\mathbf{X},\theta)$ directly from data [Zhang and Oles, 2000]. If we have a diagnostic model, the log-likelihood function of a labeled-unlabeled dataset is:

$$\begin{aligned} L(\theta) &= \sum_{i=1}^{N_l} \log p(c_i|\mathbf{x}_i,\theta) + \sum_{j=1}^{N_l} \log p(\mathbf{x}_j) \\ &+ \sum_{k=(N_l+1)}^{N_l+N_u} \log p(\mathbf{x}_k) + \log\left(\lambda^{N_l}(1-\lambda)^{N_u}\right), \end{aligned}$$

where we explicitly indicate the fact that $p(\mathbf{X})$ does not depend on θ. Any attempt to maximize this log-likelihood function with respect to θ will not be affected by unlabeled data; only the first term of the likelihood has non-zero derivative with respect to θ. Maximum likelihood estimation of diagnostic models, in the narrow sense of diagnostic models defined above, cannot process unlabeled data for *any* given dataset. These arguments regarding diagnostic models indicate that, as long as maximum likelihood estimation is the method of choice for estimation, we must resort to some form of generative model for our classifiers (Zhang and Oles develop a similar argument using the Fisher information [Zhang and Oles, 2000]). This work adopts maximum likelihood estimators and generative models.

Statistical intuition suggests that it is reasonable to expect an average improvement in classification performance for any increase in the number of samples (labeled or unlabeled): the more data, the better. In fact, it would seem that any increase in the number of samples should contribute to a reduction in the variance of $\hat{\theta}$, and a smaller variance should be beneficial to classification — this intuitive reasoning suggests that unlabeled data must be used whenever available. In Section 4.5.1, we show how this informal argument can be formalized and the circumstances in which it is valid.

Before we continue with the analysis, a number of assumptions about $P(C,\mathbf{X})$ and $P(C,\mathbf{X}|\theta)$ must be made. We will need to apply the assumptions for several variables in Section 4.5, so it is convenient to have them stated in generic form for a variable Y with distribution $P(Y)$ and for a model given by $P(Y|\theta)$.

Assumption 4.2

1 *The distribution $P(Y)$ is defined on a measurable Euclidean space with measurable Radon-Nikodym density $p(Y)$.*

2 *Distributions in the family $P(Y|\theta)$ are defined on the same measurable space as $P(Y)$, with a measurable Radon-Nikodym density $p(Y|\theta)$ for every value of θ, and continuous on θ for every value of Y.*

3 *The partial derivatives $\partial p(Y|\theta)\,/\partial\theta_i$ are measurable functions of Y for every value of θ, and continuously twice differentiable functions of θ for every value of Y.*

4 *The function $|p(Y|\theta)|$ and the functions $|\partial^2 \log p(Y|\theta)\,/\partial\theta_i\theta_j|$ and $|\partial \log p(Y|\theta)\,/\partial\theta_i \times \partial \log p(Y|\theta)\,/\partial\theta_j|$ (for every pair of components θ_i and θ_j of θ), are dominated by functions integrable with respect to $p(Y)$, for all Y and all θ.*

5 *The expected value $E[\log p(Y)]$ exists.*

6 *The expected values $E[\log p(Y|\theta)]$ exist for every θ, and each function attains a maximum at some value of θ in an open neighborhood in Θ.* □

We adopt Assumption 4.2 throughout for Y equal to $(C, \mathbf{X})$ or $\mathbf{X}$. Conditions 1 and 2 ensure that the distribution and density are well defined, conditions 3-6 ensure that the derivatives of the densities are well defined, and necessary expectations exist, allowing us to later compute a bound on the covariance matrix of the estimates of θ.

5. Asymptotic Properties of Maximum Likelihood Estimation with Labeled and Unlabeled Data

The following analysis provides a unified explanation of the behavior of classifiers trained with both labeled and unlabeled data for both cases; when the model is correct and when it is not. To do so we derive the asymptotic properties of maximum likelihood estimators for the labeled-unlabeled case.

We resort to results that were originally developed by Berk [Berk, 1966] and Huber [Huber, 1967], and then extended by White [White, 1982]. The basic results are summarized in Theorem 4.3. In this theorem and later, a Gaussian density with mean μ and variance σ^2 is denoted by $N(\mu, \sigma^2)$. The following matrices are used (matrices are formed by running through the indices i and j):

$$\begin{aligned} A_Y(\theta) &= E\left[\partial^2 \log p(Y|\theta)\,/\partial\theta_i\theta_j\right], \\ B_Y(\theta) &= E[(\partial \log p(Y|\theta)\,/\partial\theta_i)(\partial \log p(Y|\theta)\,/\partial\theta_j)]. \end{aligned}$$

THEOREM 4.3 (THEOREMS 2.2, 3.1, AND 3.2 FROM [WHITE, 1982]) *Consider a parametric model $P(Y|\theta)$ that satisfies Assumption 4.2. Consider a sequence of maximum likelihood estimates $\hat{\theta}_N$, obtained by maximization of $\sum_{i=1}^{N} \log p(y_i|\theta)$, with an increasing number of independent samples N, all identically distributed according to $P(Y)$. Then $\hat{\theta}_N \to \theta^*$ as $N \to \infty$ for θ in an open neighborhood of θ^*, where θ^* maximizes $E[\log p(Y|\theta)]$. If θ^* is interior to Θ, θ^* is a regular point of $A_Y(\theta)$ and $B_Y(\theta^*)$ is non-singular, then $\sqrt{N}\left(\hat{\theta}_N - \theta^*\right) \sim N\left(0, C(\theta^*)\right)$, where $C_Y(\theta) = A_Y(\theta)^{-1} B_Y(\theta) A_Y(\theta)^{-1}$.*

Note that Theorem 4.3 does not require the distribution $P(Y)$ to belong to the family of distributions $P(Y|\theta)$.

Consider the application of Theorem 4.3 to semi-supervised learning. Here the samples are realizations of

$$\begin{cases} (C, \mathbf{X}) & \text{with probability } \lambda; \\ \mathbf{X} & \text{with probability } (1-\lambda). \end{cases} \tag{4.6}$$

To apply Theorem 4.3, it is convenient to obtain a single expression for both situations. Denote by $\tilde{C}$ a random variable that assumes the same values of C plus the "unlabeled" value u. We have $p\left(\tilde{C} \neq u\right) = \lambda$. The actually observed samples are realizations of $(\tilde{C}, \mathbf{X})$, so we can summarize Eqn. (4.6) compactly as follows:

$$\begin{aligned} \tilde{p}\left(\tilde{C} = c, \mathbf{X} = \mathbf{x}\right) &= (\lambda p(C = c, \mathbf{X} = \mathbf{x}))^{I_{\{\tilde{C} \neq u\}}(c)} \cdot \\ &\quad ((1-\lambda) p(\mathbf{X} = \mathbf{x}))^{I_{\{\tilde{C} = u\}}(c)}, \end{aligned} \tag{4.7}$$

where $p(\mathbf{X})$ is a mixture density obtained from $p(C, \mathbf{X})$ (Eqn. (4.4)). Accordingly, the parametric model adopted for $(\tilde{C}, \mathbf{X})$ is:

$$\begin{aligned} \tilde{p}\left(\tilde{C} = c, \mathbf{X} = \mathbf{x}|\theta\right) &= (\lambda p(C = c, \mathbf{X} = \mathbf{x}|\theta))^{I_{\{\tilde{C} \neq u\}}(c)} \cdot \\ &\quad ((1-\lambda) p(\mathbf{X} = \mathbf{x}|\theta))^{I_{\{\tilde{C} = u\}}(c)}. \end{aligned} \tag{4.8}$$

Using these definitions, we obtain:

THEOREM 4.4 ([COZMAN AND COHEN, 2003]) *Consider supervised learning where samples are randomly labeled with probability λ (Eqn. (4.6)). Adopt the assumptions in Theorem 4.3, with Y replaced by $(C, \mathbf{X})$ and by $\mathbf{X}$, are valid, and also adopt Assumption 4.1 for the marginal distributions of $\mathbf{X}$. Then the value of θ^*, the limiting value of maximum likelihood estimates, is:*

$$\arg\max_{\theta} \lambda E[\log p(C, \mathbf{X}|\theta)] + (1-\lambda) E[\log p(\mathbf{X}|\theta)], \tag{4.9}$$

where the expectations are with respect to $p(C, \mathbf{X})$. Additionally, $\sqrt{N}(\hat{\theta}_N - \theta^*) \sim N(0, C_\lambda(\theta))$ as $N \to \infty$, where $C_\lambda(\theta)$ is given by:

$$C_\lambda(\theta) = A_\lambda(\theta)^{-1} B_\lambda(\theta) A_\lambda(\theta)^{-1}, \tag{4.10}$$

where

$$\begin{aligned} A_\lambda(\theta) &= \left(\lambda A_{(C,\mathbf{X})}(\theta) + (1-\lambda) A_{\mathbf{X}}(\theta)\right) \text{ and} \\ B_\lambda(\theta) &= \left(\lambda B_{(C,\mathbf{X})}(\theta) + (1-\lambda) B_{\mathbf{X}}(\theta)\right), \end{aligned}$$

evaluated at θ^*.

Proof. By Theorem 4.3, θ^* maximizes $E\left[\log \tilde{p}\left(\tilde{C}, \mathbf{X}|\theta\right)\right]$ (expectation with respect to $\tilde{p}\left(\tilde{C}, \mathbf{X}\right)$). We have:

$$\begin{aligned} E\left[\log p\left(\tilde{C}, \mathbf{X}|\theta\right)\right] &= E\Big[I_{\{\tilde{C}\neq u\}}(\tilde{C})\left(\log\lambda + \log p(C, \mathbf{X}|\theta)\right) \\ &\qquad + I_{\{\tilde{C}=u\}}(\tilde{C})\left(\log(1-\lambda) + \log p(\mathbf{X}|\theta)\right)\Big] \\ &= \lambda\log\lambda + (1-\lambda)\log(1-\lambda) + \\ & E\left[I_{\{\tilde{C}\neq u\}}(\tilde{C})\log p(C, \mathbf{X}|\theta)\right] + E\left[I_{\{\tilde{C}=u\}}(\tilde{C})\log p(\mathbf{X}|\theta)\right]. \end{aligned}$$

The first two terms of this expression are irrelevant to maximization with respect to θ. The last two terms are equal to

$$\lambda E\left[\log p(C, \mathbf{X}|\theta)\,|\tilde{C}\neq u\right] + (1-\lambda)E\left[\log p(\mathbf{X}|\theta)\,|\tilde{C}=u\right].$$

As we have $\tilde{p}\left(\tilde{C}, \mathbf{X}|\tilde{C}\neq u\right) = p(C, \mathbf{X})$ and $\tilde{p}\left(\mathbf{X}|\tilde{C}=u\right) = p(\mathbf{X})$ (Eqn. (4.6)), the last expression is equal to

$$\lambda E[\log p(C, \mathbf{X}|\theta)] + (1-\lambda)E[\log p(\mathbf{X}|\theta)],$$

where the last two expectations are now with respect to $p(C, \mathbf{X})$. Thus we obtain Expression (4.9). Expression (4.10) follows directly from White's theorem and Expression (4.9), replacing Y by C, X and $\mathbf{X}$ where appropriate.

Expression (4.9) indicates that semi-supervised learning can be viewed asymptotically as a "convex" combination of supervised and unsupervised learning. The objective function for semi-supervised learning is a combination of the objective function for supervised learning ($E[\log p(C, \mathbf{X}|\theta)]$) and the objective function for unsupervised learning ($E[\log p(\mathbf{X}|\theta)]$).

Denote by θ^*_λ the value of θ that maximizes Expression (4.9) for a given λ. Then θ^*_1 is the asymptotic estimate of θ for *supervised* learning, denoted by θ^*_l. Likewise, θ^*_0 is the asymptotic estimate of θ for *unsupervised* learning, denoted by θ^*_u.

The asymptotic covariance matrix is positive definite as $B_Y(\theta)$ is positive definite and $A_Y(\theta)$ is symmetric for any Y,

$$\theta A(\theta)^{-1}B_Y(\theta)A(\theta)^{-1}\theta^T = w(\theta)B_Y(\theta)w(\theta)^T > 0,$$

where $w(\theta) = \theta A_Y(\theta)^{-1}$. We see that asymptotically, an increase in N, the number of labeled and unlabeled samples, will lead to a reduction in the variance of $\hat{\theta}$.

Such a guarantee can perhaps be the basis for the optimistic view that unlabeled data should always be used to improve classification accuracy. In the following, we show this view is valid when the model is correct, and that it is not always valid when the model is incorrect.

5.1 Model Is Correct

Suppose first that the family of distributions $P(C, \mathbf{X}|\theta)$ contains the distribution $P(C, \mathbf{X})$; that is, $P(C, \mathbf{X}|\theta_\top) = P(C, \mathbf{X})$ for some $\theta_\top$. Under this condition, the maximum likelihood estimator is consistent, thus, $\theta_l^* = \theta_u^* = \theta_\top$ given identifiability. Thus, $\theta_\lambda^* = \theta_\top$ for any $0 \leq \lambda \leq 1$.

Additionally, using White's results [White, 1982], $A(\theta_\lambda^*) = -B(\theta_\lambda^*) = \mathbf{I}(\theta_\lambda^*)$, where $\mathbf{I}()$ denotes the Fisher information matrix. Thus, the Fisher information matrix can be written as:

$$\mathbf{I}(\theta) = \lambda \mathbf{I}_l(\theta) + (1 - \lambda)\mathbf{I}_u(\theta), \tag{4.11}$$

which matches the derivations made by Zhang and Oles [Zhang and Oles, 2000]. The significance of Expression (4.11) is that it allows the use of the Cramer-Rao lower bound (CRLB) on the covariance of a consistent estimator:

$$\mathrm{Cov}(\hat{\theta}_N) \geq \frac{1}{N}(\mathbf{I}(\theta))^{-1} \tag{4.12}$$

where N is the number of data (both labeled and unlabeled) and $\mathrm{Cov}(\hat{\theta}_N))$ is the estimator's covariance matrix with N samples.

Consider the Taylor expansion of the classification error around $\theta_\top$, as suggested by Shahshahani and Landgrebe [Shahshahani and Landgrebe, 1994a], linking the decrease in variance associated with unlabeled data to a decrease in classification error, and assuming existence of necessary derivatives:

$$\mathbf{e}(\hat{\theta}) \approx \mathbf{e}_{\mathsf{B}} + \left.\frac{\partial \mathbf{e}(\theta)}{\partial \theta}\right|_{\theta_\top} \left(\hat{\theta} - \theta_\top\right) + \frac{1}{2}\mathrm{tr}\left(\left.\frac{\partial^2 \mathbf{e}(\theta)}{\partial \theta^2}\right|_{\theta_\top} \left(\hat{\theta} - \theta_\top\right) \left(\hat{\theta} - \theta_\top\right)^T \right). \tag{4.13}$$

Take expected values on both sides. Asymptotically, the expected value of the second term in the expansion is zero, as maximum likelihood estimators are asymptotically unbiased when the model is correct. Shahshahani and Landgrebe thus argue that

$$E\left[\mathbf{e}(\hat{\theta})\right] \approx \mathbf{e}_{\mathsf{B}} + (1/2)\mathrm{tr}\left((\partial^2 \mathbf{e}(\theta)/\partial \theta^2)|_{\theta_\top} \mathrm{Cov}(\hat{\theta}) \right)$$

where $\mathrm{Cov}(\hat{\theta})$ is the covariance matrix for $\hat{\theta}$. They show that if $\mathrm{Cov}(\theta') \geq \mathrm{Cov}(\theta'')$ for some θ', θ'', then the second term in the approximation is larger for θ' than for θ''. And again, because $\mathbf{I}_u(\theta)$ is always positive definite, $\mathbf{I}_l(\theta) \leq$

$\mathbf{I}(\theta)$. Thus, using the CRLB (Eqn. 4.12), the covariance with labeled and unlabeled data is smaller than the covariance with just labeled data, leading to the conclusion that *unlabeled data must cause a reduction in classification error when the model is correct.* It should be noted that this argument holds as the number of records goes to infinity, and is an approximation for finite values.

A more formal, but less general, argument is presented by Ganesalingam and McLachlan [Ganesalingam and McLachlan, 1978] as they compare the relative efficiency of labeled and unlabeled data. Castelli also derives a Taylor expansion of the classification error, to study estimation of the mixing factors, $p(C = c)$; the derivation is very precise and states all the required assumptions [Castelli, 1994].

5.2 Model Is Incorrect

We now study the more realistic scenario where the distribution $P(C, \mathbf{X})$ does not belong to the family of distributions $P(C, \mathbf{X}|\theta)$. In view of Theorem 4.4, it is perhaps not surprising that unlabeled data can have the deleterious effect discussed in Section 4.3. Suppose that $\theta^*_u \neq \theta^*_l$ and that $\mathbf{e}(\theta^*_u) > \mathbf{e}(\theta^*_l)$, as in the examples in the next section.[3] If we observe a large number of labeled samples, the classification error is approximately $\mathbf{e}(\theta^*_l)$. If we then collect more samples, most of which unlabeled, we eventually reach a point where the classification error approaches $\mathbf{e}(\theta^*_u)$. So, the net result is that we started with classification error close to $\mathbf{e}(\theta^*_l)$, and by adding a great number of unlabeled samples, classification performance degraded. The basic fact here is that estimation and classification bias are affected differently by different values of λ. Hence, a necessary condition for this kind of performance degradation is that $\mathbf{e}(\theta^*_u) \neq \mathbf{e}(\theta^*_l)$; a sufficient condition is that $\mathbf{e}(\theta^*_u) > \mathbf{e}(\theta^*_l)$.

The focus on asymptotics is adequate as we want to eliminate phenomena that can vary from dataset to dataset. If $\mathbf{e}(\theta^*_l)$ is smaller than $\mathbf{e}(\theta^*_u)$, then a large enough labeled dataset can be dwarfed by a much larger unlabeled dataset — the classification error using the whole dataset can be larger than the classification error using the labeled data only.

As a digression, note that the asymptotic estimate θ^*_λ is obtained as the solution of an equation of the form $g(\lambda, \theta) = \lambda f_1(\theta) + (1-\lambda) f_2(\theta) = 0$ (f_1 and f_2 are derivatives of expectations with respect to θ). Given suitable regularity assumptions, $E[\log p(C, \mathbf{X}|\theta)]$ and $E[p(\mathbf{X}|\theta)]$ are continuous and differentiable

[3] We have to handle a difficulty with $\mathbf{e}(\theta^*_u)$: given only unlabeled data, there is no information to decide the labels for decision regions, and then the classification error is 1/2 [Castelli, 1994]. Instead of actually using $\mathbf{e}(\theta^*_u)$, we could consider $\mathbf{e}(\theta^*_\epsilon)$ for any value of $\epsilon > 0$. To simplify the discussion, we avoid the complexities of $\mathbf{e}(\theta^*_\epsilon)$ by assuming that, when $\lambda = 0$, an "oracle" will be available to indicate the labels of the decision regions.

functions of θ. Then the partial derivatives of $g(\lambda, \theta)$ with respect to λ and θ are continuous, and the derivative of $g(\lambda, \theta)$ with respect to θ is not zero at θ^*_λ except in cases where $A(\theta^*_\lambda)$ or $B(\theta^*_\lambda)$ have pathological behavior (such behavior could happen with unlabeled data [Redner and Walker, 1984]). Barring those cases, the implicit function theorem then guarantees that θ^*_λ is a continuous function of λ. This shows that the "path" followed by the solution is a continuous one, as also assumed by Corduneanu and Jaakkola in their discussion of numerical methods for semi-supervised learning [Corduneanu and Jaakkola, 2002].

A geometric interpretation of the "path" followed by the solution can be derived from the fact that Eq. (4.9) can be reformulated to show that finding the optimal parameters θ^*_λ is equivalent to minimizing the KL divergence (distance) between the true joint distribution $p(C, X)$ and the model of the mixture density given in Eq. (4.7). With only labeled data ($\lambda = 1$), the KL distance being minimized is[4]:

$$D(p(C, \mathbf{X})\,||p(C, \mathbf{X}|\theta)) = \sum_{\mathbf{X},C} p(\mathbf{X}, C) \log \frac{p(C, \mathbf{X})}{p(C, \mathbf{X}|\theta)}, \tag{4.14}$$

while with unlabeled data ($\lambda = 0$) the KL distance minimized is given by:

$$D(p(\mathbf{X})\,||p(\mathbf{X}|\theta)) = \sum_{\mathbf{X}} p(\mathbf{X}) \log \frac{\sum_C p(C, \mathbf{X})}{\sum_C p(C, \mathbf{X}|\theta)}. \tag{4.15}$$

We see that with unlabeled data, the KL distance between the marginal of the true distribution and the marginal of the family of distributions is minimized, while with labeled data, the distance between the joint distributions is minimized. For λ between 0 and 1, the solution is a distribution between these two extreme solutions. Because the model is incorrect, minimizing the first distance does not necessarily minimize the latter. Figure 4.1 illustrates this geometric interpretation of the path, where the distance shown represents the KL divergence.

The KL divergence interpretation can also be used to further explain why it is that estimates with labeled data are typically better classifiers than estimates with unlabeled data, i.e., $\mathbf{e}(\theta^*_l) < \mathbf{e}(\theta^*_u)$. First we note that the classification bias (denoted as $b(\theta)$), for a binary classification problem, is upper bounded by

[4] Summation can be replaced by integration when $\mathbf{X}$ is continuous.

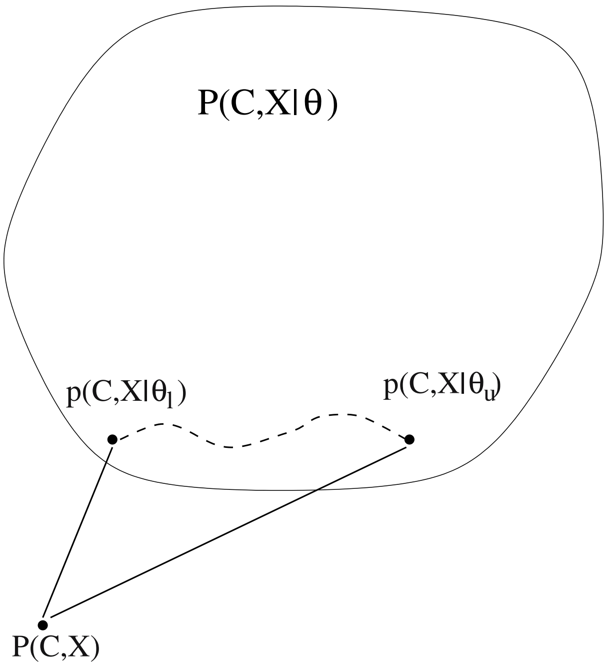


Figure 4.1. Geometric interpretation of the path between labeled and unlabeled maximum likelihood based treating KL divergence as distance. The true joint distribution is outside the space of achievable distributions based on the modeling assumptions.

the L_1 norm as follows [Devroye et al., 1996]:

$$\begin{aligned} b(\theta) &= \mathbf{e}(\theta) - \mathbf{e}_{\mathsf{B}} \\ &= 2\int |p(c=1|\mathbf{x}) - 1/2| I_{[g(\mathbf{x};\theta)\neq g^*(\mathbf{x})]} p(\mathbf{x})\, dx \\ &\leq 2\int |p(c=1|\mathbf{x}) - p(c=1|\mathbf{x},\theta)\,| p(\mathbf{x})\, dx \\ &= 2E_{\mathbf{X}}[|p(c=1|\mathbf{x}) - p(c=1|\mathbf{x},\theta)\,|]. \end{aligned}$$

The L_1 norm is further upper bounded by the KL divergence between the distributions. Using and Equations (4.14) and (4.15) we conclude that:

$$D(p(C|\mathbf{X})\,||p(C|\mathbf{X},\theta_l^*)) \leq D(p(C|\mathbf{X})\,||p(C|\mathbf{X},\theta_u^*)). \tag{4.16}$$

Thus, the bound on the L_1 norms is smaller for the labeled classifier, which might lead to a smaller bound on the classification bias, leading to a smaller classification error. Although the last part of the argument is not guaranteed, it could give the first indication on why estimates based on unlabeled data are the ones that typically yield classifiers with a larger classification error than those trained with labeled data. The examples in the following section prove that such a possibility indeed exists for different examples.

5.3 Examples: Unlabeled Data Degrading Performance with Discrete and Continuous Variables

The previous discussion alluded to the possibility that $\mathbf{e}(\theta_u^*) > \mathbf{e}(\theta_l^*)$ when the model is incorrect. To the skeptical reader who still may think that this will not occur in practice, or that numerical instabilities are to blame, we analytically show how this occurs with several examples of obvious practical significance.

EXAMPLE 4.5 *Consider the following (fictitious) classification problem. We are interested in predicting a baby's gender (G $=$ Boy or Girl) at the 20'th week of pregnancy based on two attributes: whether the mother craved chocolate in the first trimester (Ch $=$ Yes or No), and whether the mother's weight gain was more or less than 15lbs (W $=$ $More$ or $Less$). Suppose that the true underlying joint distribution, $p(G, Ch, W)$ can be represented with the following graph: G $\rightarrow$ Ch $\rightarrow$ W (i.e., $W \perp\!\!\!\perp G|Ch$) and the values of the conditional probabilities are specified as: $p(G = Boy) = 0.5$, $p(Ch = No|G = Boy) = 0.1$, $p(Ch = No|G = Girl) = 0.8$, $p(W = Less|Ch = No) = 0.7$, $p(W = Less|Ch = Yes) = 0.2$. With these probabilities we compute the a-posteriori probability of G (which depends only on Ch):*

$p(G\|Ch)$	*Girl*	*Boy*	*Prediction*
No	*0.89*	*0.11*	*Girl*
Yes	*0.18*	*0.82*	*Boy*

The Bayes error rate for this problem can be easily computed and found to be 15%. Suppose that we incorrectly assume the following (Naive Bayes) relationship between the variables: $Ch \leftarrow G \rightarrow W$, thus we incorrectly assume that weight gain is independent of chocolate craving given the gender. Suppose also that we are given the values for $p(Ch|G)$ and we wish to estimate $p(G)$ and $p(W|G)$ from data. We use Eq.(4.9) to get both the estimates with infinite labeled data ($\lambda = 1$) and the estimates with infinite unlabeled data ($\lambda = 0$). For the labeled case, $\hat{p}(G)$ is exactly 0.5. The estimate of $\hat{p}(W|G)$ is $p(W|G)$ computed from the true distribution: $\hat{p}(W = Less|G = Girl) = 0.6$, $\hat{p}(W = Less|G = Boy) = 0.25$, leading to the a-posteriori probability of G:

$\hat{p}(G\|Ch, W)$	*Girl*	*Boy*	*Prediction*
No, Less	*0.95*	*0.05*	*Girl*
No, More	*0.81*	*0.19*	*Girl*
Yes, Less	*0.35*	*0.65*	*Boy*
Yes, More	*0.11*	*0.89*	*Boy*

We see that although there is a non-zero bias between the estimated distribution and the true distribution, the prediction remains unchanged. Thus, there is no increase in classification error compared to the Bayes error rate. The solution for the unlabeled case involves solving a system of three equations with three variables (using the marginal, $p(Ch, W)$ from the true distribution), yielding the following estimates: $\hat{p}(G = Boy) = 0.5$, $\hat{p}(W = Less|G = Girl) = 0.78$, $\hat{p}(W = Less|G = Boy) = 0.07$, with a-posteriori probability:

$\hat{p}(G\|Ch, W)$	*Girl*	*Boy*	*Prediction*
No, Less	*0.99*	*0.01*	*Girl*
No, More	*0.55*	*0.45*	*Girl*
Yes, Less	*0.71*	*0.29*	**Girl**
Yes, More	*0.05*	*0.95*	*Boy*

Here we see that the prediction is changed from the optimal in the case of $Ch = Yes, W = Less$; instead of predicting $G = Boy$, we predict Girl. We also see that the bias is further increased, compared to the labeled case. We can easily find the expected error rate to be at 22%, an increase *of 7% in error.*□

For the second example, we will assume that bivariate Gaussian samples (X, Y) are observed. The only modeling error is an ignored dependency between observables. This type of modeling error is quite common in practice and has been studied in the context of supervised learning [Ahmed and Lachenbruch, 1977; McLachlan, 1992]. It is often argued that ignoring some dependencies can be a positive decision, as we may see a reduction in the number of parameters to be estimated and a reduction on the variance of estimates [Friedman, 1997].

EXAMPLE 4.6 *Consider real-valued observations (X, Y) taken from two classes c' and c''. We know that X and Y are Gaussian variables, and we know their means and variances given the class C. The mean of (X, Y) is $(0, 3/2)$ conditional on $\{C = c'\}$, and $(3/2, 0)$ conditional on $\{C = c''\}$. Variances for X and for Y conditional on C are equal to 1. We do not know, and have to estimate, the mixing factor $\eta = p(C = c')$. The data is sampled from a distribution with mixing factor equal to 3/5.*

We want to obtain a Naive-Bayes classifier that can approximate $p(C|X, Y)$; Naive-Bayes classifiers are based on the assumption that X and Y are independent given C. Suppose that X and Y are independent conditional on $\{C = c'\}$ but that X and Y are dependent conditional on $\{C = c''\}$. This dependency is manifested by a correlation $\rho = E[(X - E[X])(Y - E[Y])] = 4/5$. If we knew the value of ρ, we would obtain an optimal classification boundary on the plane $X \times Y$. This optimal classification boundary is shown *in Figure 4.2, and is defined by the function*

$$y = \left(40x - 87 + \sqrt{5265 - 2160x + 576x^2 + 576\log(100/81)}\right)/32.$$

Under the incorrect assumption that $\rho = 0$*, the classification boundary is then linear:*

$$y = x + 2\log((1-\hat{\eta})/\hat{\eta})/3,$$

and consequently it is a decreasing function of $\hat{\eta}$*. With labeled data we can easily obtain* $\hat{\eta}$ *(a sequence of Bernoulli trials); then* $\eta_l^* = 3/5$ *and the classification boundary is given by* $y = x - 0.27031$.

Note that the (linear) boundary obtained with labeled data is not the best possible linear boundary. We can in fact find the best possible linear boundary of the form $y = x + \gamma$*. For any* γ*, the classification error* $\mathbf{e}(\gamma)$ *is*

$$\begin{aligned}\mathbf{e}(\gamma) &= \frac{3}{5}\int_{-\infty}^{\infty}\int_{-\infty}^{x+\gamma} N\left(\begin{bmatrix}0\\3/2\end{bmatrix}, \mathit{diag}[1,1]\right) dydx \\ &+ \frac{2}{5}\int_{-\infty}^{\infty}\int_{x+\gamma}^{\infty} N\left(\begin{bmatrix}3/2\\0\end{bmatrix}, \begin{bmatrix}1 & 4/5\\4/5 & 1\end{bmatrix}\right) dydx.\end{aligned}$$

By interchanging differentiation with respect to γ *with integration, it is possible to obtain* $d\mathbf{e}(\gamma)/d\gamma$ *in closed form. The second derivative* $d^2\mathbf{e}(\gamma)/d\gamma^2$ *is positive when* $\gamma \in [-3/2, 3/2]$*; consequently there is a single minimum that can be found by solving* $d\mathbf{e}(\gamma)/d\gamma = 0$*. We find the minimizing* γ *to be* $(-9 + 2\sqrt{45/4 + \log(400/81)})/4 \approx -0.45786$*. The line* $y = x - 0.45786$ *is the best linear boundary for this problem. If we consider the set of lines of the form* $y = x + \gamma$*, we see that the farther we go from the best line, the larger the classification error. Figure 4.2 shows the linear boundary obtained with labeled data and the best possible linear boundary. The boundary from labeled data is "above" the best linear boundary.*

Now consider the computation of η_u^**, the asymptotic estimate with unlabeled data:*

$$\begin{aligned}\eta_u^* &= \arg\max_{\eta\in[0,1]}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\log\left(\eta N([0,3/2]^T, \mathit{diag}[1,1]) + (1-\eta)N([3/2,0]^T, \mathit{diag}[1,1])\right) \\ &\cdot\left((3/5)N([0,3/2]^T, \mathit{diag}[1,1]) + (2/5)N\left(\begin{bmatrix}3/2\\0\end{bmatrix}, \begin{bmatrix}1 & 4/5\\4/5 & 1\end{bmatrix}\right)\right) dydx.\end{aligned}$$

The second derivative of this double integral is always negative (as can be seen interchanging differentiation with integration), so the function is concave and there is a single maximum. We can search for the zero of the derivative of the double integral with respect to η*. We obtain this value numerically,* $\eta_u^* \approx 0.54495$*. Using this estimate, the linear boundary from unlabeled data is* $y = x - 0.12019$*. This line is "above" the linear boundary from labeled data, and, given the previous discussion, leads to a larger classification error* *than the boundary from unlabeled data. We have:* $\mathbf{e}(\gamma) = 0.06975$*;* $\mathbf{e}(\theta_l^*) = 0.07356$*;* $\mathbf{e}(\theta_u^*) = 0.08141$*. The boundary obtained from unlabeled data is also shown in Figure 4.2.* □

This example suggests the following situation. Suppose we collect a large number N_l of labeled samples from $p(C, X)$, with $\eta = 3/5$ and $\rho = 4/5$. The labeled estimates form a sequence of Bernoulli trials with probability $3/5$, so the estimates quickly approach η_l^* (the variance of $\hat{\eta}$ decreases as $6/(25N_l)$). If we add a very large amount of unlabeled data to our data, $\hat{\eta}$ approaches η_u^* and the classification error increases.

By changing the "true" mixing factor and the correlation ρ, we can create different situations. For example, if $\eta = 3/5$ and $\rho = -4/5$, the best linear boundary is $y = x - 0.37199$, the boundary from labeled data is $y = x - 0.27031$, and the boundary from unlabeled data is $y = x - 0.34532$; the latter boundary is "between" the other two — additional unlabeled data lead to improvement in classification performance. As another example, if $\eta = 3/5$ and $\rho = -1/5$, the best linear boundary is $y = x - 0.29044$, the boundary from labeled data is $y = x - 0.27031$, and the boundary from unlabeled data is $y = x - 0.29371$. The best linear boundary is "between" the other two. We in fact attain the best possible linear boundary by mixing labeled and unlabeled data with $\lambda = 0.08075$.

5.4 Generating Examples: Performance Degradation with Univariate Distributions

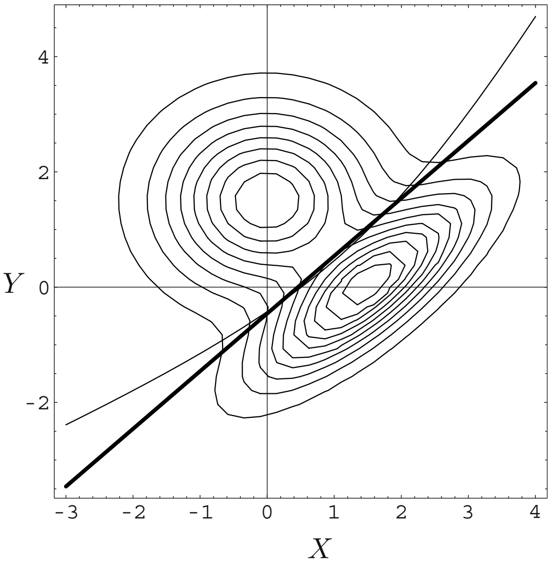
It might be thought that Examples 4.6 and 4.5 display a rare combination of carefully adjusted parameters. This is certainly not the case, as we have produced several examples with similar behavior. It is actually interesting to discuss a heuristic method to produce such examples, and to illustrate the method with an univariate example.

Consider the following procedure to generate situations where $\mathbf{e}(\theta_u^*) > \mathbf{e}(\theta_l^*)$:

1 Take some set of distributions parameterized by θ in Θ, such that all distributions satisfy Assumptions 4.1 and 4.2.

2 Select a value $\theta_l \in \Theta$. Then find the value $\theta_u \in \Theta$ such that θ_u produces the worst possible classification error when used to classify samples from $p(C, \mathbf{X}|\theta_l)$. If $p(C, \mathbf{X}|\theta_l)$ and $p(C, \mathbf{X}|\theta_u)$ produce identical classification errors, then enlarge the set of distributions and start again.

3 Define $p_n(C, \mathbf{X}) = p(C, \mathbf{X}|\theta_l) \times p(\mathbf{X}|\theta_u)$. Obtain the value θ_a that maximizes the expected value $E_{p_n}[\log p(C, \mathbf{X}|\theta)]$. If $\mathbf{e}(\theta_a)$ is the same as $\mathbf{e}(\theta_u)$, start again by selecting a different θ_l or by modifying the set of distributions parameterized by θ.



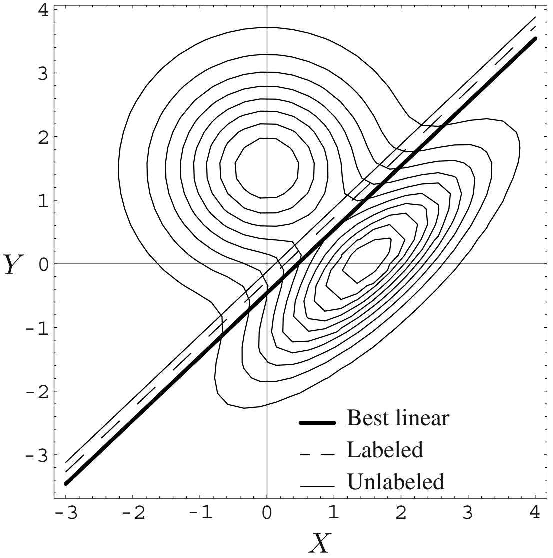


Figure 4.2. Graphs for Example 4.6. On the top, contour plots of the mixture $p(X, Y)$, the optimal classification boundary (quadratic curve) and the best possible classification boundary of the form $y = x + \gamma$. On the bottom, the same contour plots, and the best linear boundary (lower line), the linear boundary obtained from labeled data (middle line) and the linear boundary obtained from unlabeled data (upper line); thus the classification error of the unlabeled classifier is larger than that of the labeled classifier.

Note that $p_n(C, \mathbf{X})$ cannot induce a distribution parameterized by Θ, as otherwise identifiability would be violated ($p(\mathbf{X}|\theta_u)$ would correspond to at least two different distributions). If a classifier is learned with unlabeled data sampled from $p_n(C, \mathbf{X})$, by construction the classification error is the worst possible ($\mathbf{e}(\theta_u)$); if a classifier is learned from labeled data, then by Theorem 4.3 the result is $p(C, \mathbf{X}|\theta_a)$ and by construction $\mathbf{e}(\theta_a) < \mathbf{e}(\theta_u)$. The next example was generated using such a procedure[5].

EXAMPLE 4.7 *Consider real-valued observations X taken from two classes c' and c''. We adopt a mixture of known Gaussians as our model, and we know that the mixing factor η belongs to the interval* $[0.2, 0.8]$. *The exact value of η must be estimated. We adopt $p(X|C = c') = N(0, 1)$ and $p(X|C = c'') = N(3, 1)$.*

The optimal "plug-in" classification rule depends on the estimate $\hat{\eta}$ as follows:

$$\{c' \textit{ if } X \leq b(\hat{\eta}); c'' \textit{ otherwise}\}, \quad \textit{where } b(\hat{\eta}) = (9 - 2\log(1/\hat{\eta} - 1))/6.$$

Suppose the observations are actually sampled from

$$\begin{aligned} p_n(C = c', X) &= p_n(C = c'|X)p_n(X) \\ &= \frac{0.8N(0,1)}{0.8N(0,1) + 0.2N(3,1)}(0.2N(0,1) + 0.8N(3,1)), \\ p_n(C = c'', X) &= p_n(C = c''|X)p_n(X) \\ &= \frac{0.2N(3,1)}{0.8N(0,1) + 0.2N(3,1)}(0.2N(0,1) + 0.8N(3,1)). \end{aligned}$$

While the joint distribution $p_n(C, X)$ is not a mixture of Gaussians, the marginal $p_n(X)$ is a mixture of Gaussians with mixing factor 0.2. Likewise, the posterior $p_n(C|X)$ is identical to the posterior of a mixture of Gaussians with mixing factor 0.8. Note that the classification error for $\hat{\eta}$ is $\mathbf{e}(\hat{\eta}) = \int_{b(\hat{\eta})}^{\infty} p_n(C = c', X)dx + \int_{-\infty}^{b(\hat{\eta})} p_n(C = c'', X)dx$, a decreasing function from 0.2 to 0.8 (Figure 4.3). If $\hat{\eta} = 0.8$, the Bayes error is attained. The estimate with fully unlabeled data is $\eta_u^* = 0.2$, which yields the worst performance. For labeled data, we estimate η using a sequence of Bernoulli trials with probability $p_n(C = c') = \int p_n(C = c', X)dx \approx 0.339171$, so asymptotically we have $\eta_l^* \approx 0.339171$. Thus, the classification error produced with labeled data is smaller than the error produced with unlabeled data.□

[5] The construction method and the following example were developed by Fabio Cozman.

Using Theorem 4.4, we can study the behavior of classification error for varying amounts of unlabeled data. In Example 4.7, we have:

$$\begin{aligned}
\eta^*_\lambda &= \arg \max_{\eta\in[0.2,0.8]} \lambda E\Big[\log\Big(p_n(C=c',X)^{I_{\{C=c'\}}(C)} p_n(C=c'',X)^{I_{\{C=c''\}}(C)}\Big)\Big] \\
&\quad +(1-\lambda)E[\log p_n(X)] \\
&= \arg \max_{\eta\in[0.2,0.8]} \lambda\left(\int p_n(c',X)\log(\eta N(0,1))dx + \int p_n(c'',X)\log((1-\eta)N(3,1))dx\right) \\
&\quad +(1-\lambda)\int p_n(X)\log(\eta N(0,1)+(1-\eta)N(3,1))dx \\
&= \arg \max_{\eta\in[0.2,0.8]} \lambda\log\eta\times\int\frac{0.8N(0,1)}{0.8N(0,1)+0.2N(3,1)}(0.2N(0,1)+0.8N(3,1))dx \\
&\quad +\lambda\log(1-\eta)\times\int\frac{0.2N(3,1)}{0.8N(0,1)+0.2N(3,1)}(0.2N(0,1)+0.8N(3,1))dx \\
&\quad +(1-\lambda)\int(0.2N(0,1)+0.8N(3,1))\log(\eta N(0,1)+(1-\eta)N(3,1))dx \\
&= \arg \max_{\eta\in[0.2,0.8]} \lambda(\eta^*_l\log\eta+(1-\eta^*_l)\log(1-\eta)) \\
&\quad +(1-\lambda)\int(0.2N(0,1)+0.8N(3,1))\log(\eta N(0,1)+(1-\eta)N(3,1))dx.
\end{aligned}$$

The derivative (with respect to η) of the quantity to be maximized is

$$\begin{aligned}
d(\eta) &= \lambda\left(\frac{\eta^*_l}{\eta}-\frac{1-\eta^*_l}{1-\eta}\right) \\
&\quad +(1-\lambda)\int(0.2N(0,1)+0.8N(3,1))\frac{N(0,1)-N(3,1)}{\eta N(0,1)+(1-\eta)N(3,1)}dx. \qquad (4.17)
\end{aligned}$$

Figure 4.3 shows three graphs of Expression (4.16) for different values of λ. For $\lambda=1$, we have the function for labeled data only, with zero at η^*_l. For $\lambda=0$, we have the function for unlabeled data only (obtained by numerical integration), with zero at η^*_u. Any value of λ between 0 and 1 will correspond to a curve that is the "weighted" average of the other curves, with a zero between η^*_u and η^*_l, and then $\mathbf{e}(\eta^*_l)<\mathbf{e}(\eta^*_\lambda)<\mathbf{e}(\eta^*_u)$. The figure shows the curve for $\lambda=1/2$.

In many aspects, this example has the same structure as Example 4.6. In both examples, the estimates from labeled data are simple Bernoulli trials, while the estimates from unlabeled data have more complex behavior. In both examples the estimates move from θ^*_l to θ^*_u as λ goes from 1 to 0.

5.5 Distribution of Asymptotic Classification Error Bias

The examples above illustrated how unlabeled data can degrade the performance of classifiers asymptotically using specific examples. To illustrate the differences in error bias over a wider range of classification problems, we performed another experiment, simulating the case of infinite labeled data and infinite unlabeled data with a large number of different classifiers and incorrect modeling assumptions.We generated 100 different classifiers, each with

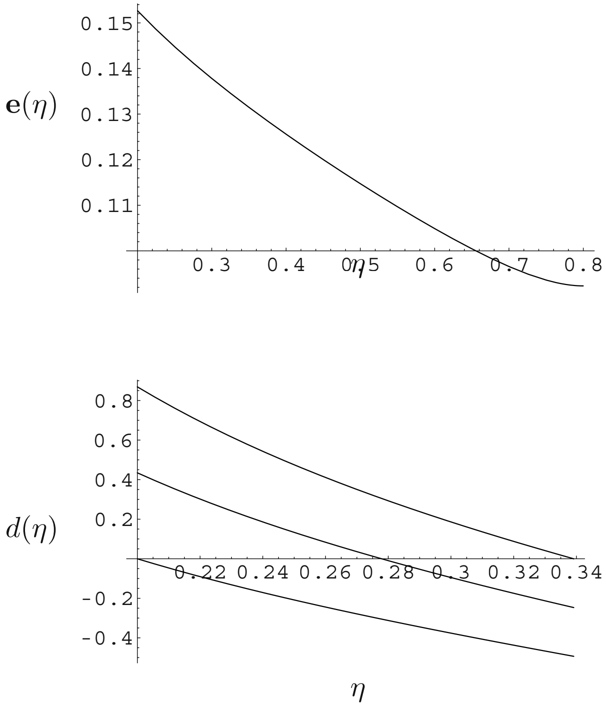


Figure 4.3. Graphs for Example 4.7. On the top, classification error $\mathbf{e}(\hat{\eta})$. On the bottom, derivatives of the function to be maximized when computing η^*_λ, for three values of λ: top curve is for $\lambda = 1$, bottom curve is for $\lambda = 0$, and curve in the middle for $\lambda = 1/2$.

two classes and four Gaussian features that are not independent of each other. The parameters of each classifier are the class prior, $\eta = p(C = 0)$, mean vectors, $\mu_0, \mu_1 \in \Re^4$ and a common covariance matrix $S \in \Re^{4x4}$. The Bayes error rate of the classifiers ranged from $0.7 - 35\%$, with most being around 10%.

For each classifier we look at 11 combinations of making incorrect independence assumptions, by assuming that features are independent of each other (from one to all features being independent of each other; overall 11 combinations). For example, if we assume that x_1 and x_3 are independent of the rest of the features, the covariance matrix under this assumption must have the form:

$$\hat{S} = \begin{pmatrix} s_{11} & 0 & 0 & 0 \\ 0 & s_{22} & 0 & s_{24} \\ 0 & 0 & s_{33} & 0 \\ 0 & s_{42} & 0 & s_{44} \end{pmatrix}.$$

For each combination we computed the classification error of two classifiers (trained under the independence assumptions); one trained with infinite labeled data and a second trained with infinite unlabeled data. With labeled data, we use the exact values that are not zero from the original covariance matrix. With unlabeled data, we approximate infinity with $100,000$ training records (which is very large compared to 25, the largest number of parameters estimated in the experiments). We use EM to learn with unlabeled data, with the starting point being the parameter set of the labeled only classifier, therefore assuring that the difference in the results of the two classifiers do not depend on the starting point of EM.

Figure 4.4 shows the histograms of the classification bias of the classifiers with incorrect assumptions for learning with labeled and unlabeled data. The histograms show that the classification bias of the labeled based classifiers tends to be more highly concentrated closer to 0 compared to the unlabeled based classifiers. In these experiments, we also observe that using unlabeled data always resulted in a higher error rate compared to using labeled data, as can be seen in Figure 4.5. The only exception is when we do not make any incorrect independence assumptions, in which the classifiers trained with unlabeled data achieved the Bayes error rate. What we understand from these histograms is that when training with labeled data, many classifiers will perform well (although might not achieve the optimal Bayes rate). However, classifiers trained with unlabeled data need to be more accurate in their modeling assumptions in order to achieve good performance and they are a great deal more sensitive to such inaccuracies.

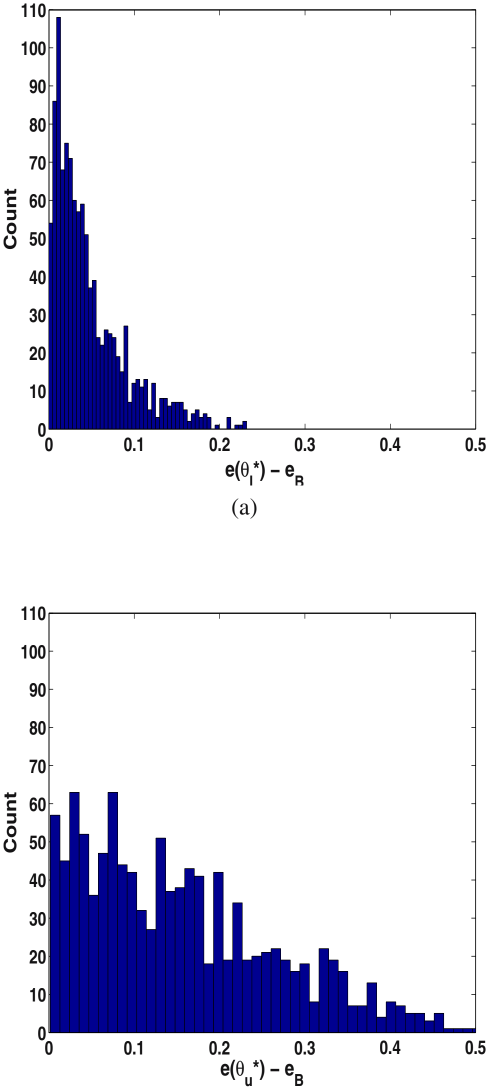
5.6 Short Summary

To summarize the results so far, we can say the following:

- Labeled and unlabeled data contribute to a reduction in variance in semi-supervised learning under maximum likelihood estimation.

- When the model is correct, the maximum likelihood estimator is consistent and both labeled and unlabeled data contribute to a reduction in classification error by reducing variance. Also, unlabeled data suffice to define the decision regions and labeled data can be used solely to label the regions (Algorithm M).

- When the model is incorrect, there may be different asymptotic estimation bias for different values of λ. Asymptotic classification error may also be different for different values of λ. An increase in the number of unlabeled samples may lead to a larger estimation bias and a larger classification error. The examples illustrated this possibility.



(a)

(b)

Figure 4.4. Histogram of classification of error bias from the Bayes error rate under incorrect independence assumptions for (a) training with labeled data and (b) training with unlabeled data.

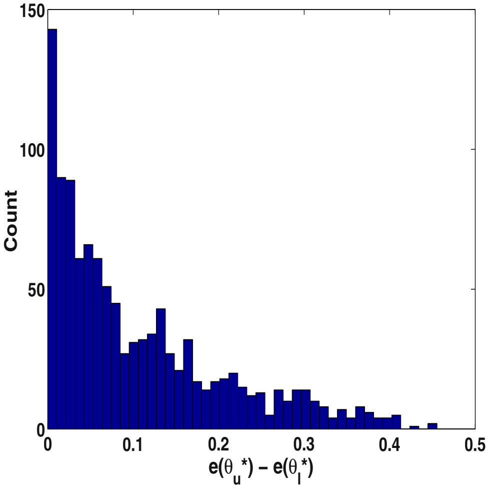


Figure 4.5. Histogram of difference between classification error rates of classifiers trained under incorrect independence with labeled data and with unlabeled data ($\mathbf{e}(\theta_u^*) - \mathbf{e}(\theta_l^*)$).

- Arguments using bounds on the classification bias, and the experiments with multiple classifiers suggests that the asymptotic classification error with unlabeled data is typically higher than with labeled data.

In essence, semi-supervised learning displays an odd failure of robustness: for certain modeling errors, more unlabeled data can degrade classification performance. Estimation bias is the central factor in this phenomenon, as the level of bias depends on the ratio of labeled to unlabeled samples. Most existing theoretical results on semi-supervised learning are based on the assumption of no modeling error, and consequently bias has not been an issue so far.

6. Learning with Finite Data

So far, we discussed the asymptotic properties of ML estimation. As in reality we never have infinite training data, a natural question is what occurs with finite training data? To illustrate the different situations that can occur we performed extensive experiments, using artificial and real data, learning correct and incorrect models with various sizes of training data (both labeled and unlabeled).

Before we describe the experiments, we note the method used to find the maximum likelihood solution with labeled and unlabeled data. When unlabeled data are present in a finite data set D, it is in general not possible to find an analytical solution that maximizes the likelihood function (Eq.(4.5)). We therefore must resort to numerical approaches. One of the most popular

Box 5.1 (EM Algorithm)

Given that Y is the set of missing variables and Z are the observed variables follow the following steps:

Initialization Initialize to θ^0. Repeat the following two steps until convergence:

Step 1: Expectation Compute the distribution $p^t(y) = p\big(y|z, \theta^{t-1}, D\big)$ over all values of Y.

Step 2: Maximization Set θ^t to the θ that maximizes the following expectation: $E_{p^t(Y)}[log(p(y, z|\theta))]$.

In semi-supervised learning, given that the class variable, C, has missing values and the features $\mathbf{X}$ have no missing values, EM follows the following steps:

Initialization: Initialize to θ^0. Repeat the following two steps until convergence:

Step 1: Expectation Compute the distribution $p\big(C = c|\mathbf{X} = \mathrm{x}|\theta^{t-1}\big)$ for every value of C and every record in D.

Step 2: Maximization Set θ^t to the θ that maximizes

$$E_{p(C|\mathbf{x},\theta^{t-1})}[log(p(c, \mathbf{x}|\theta))].$$

approaches is the expectation-maximization (EM) algorithm [Dempster et al., 1977]. EM is an iterative algorithm and is described in Box 5.1.

6.1 Experiments with Artificial Data

In our experiments, we focus on modeling assumptions related to the dependencies and independencies among the different features. Borrowing the term from Bayesian networks theory, we call these *structure* assumptions. Other types of modeling assumptions, such as the parametric form of the distribution of the features, the number of classes or the number of values of a discrete feature, are not changed and assumed to be known.

We generated datasets using two different model structures, Naive-Bayes (NB) and Tree-Augmented-Naive Bayes (TAN) [Friedman et al., 1997], varying the number of features, the number of values per feature (all features are discrete) and the size of the datasets, with different proportions of labeled and unlabeled records in each set. In the TAN structure, the class node has no parents and each feature has the class node as a parent and at most one other feature, such that the result is a tree structure for the features. This type of representation provides pairwise dependencies between the linked features. The decomposition of the joint probability distribution as represented by the TAN classifier is given as:

$$p(C, X_1, X_2, .., X_p) = p(C) \prod_{i=1}^{p} p(X_i|C, Pa_i), \tag{4.18}$$

where Pa_i is either an empty set or is equal to $X_j, i \neq j$.

The full description of the experiments, with their results are given in [Cozman and Cohen, 2001], here we introduce the main results.

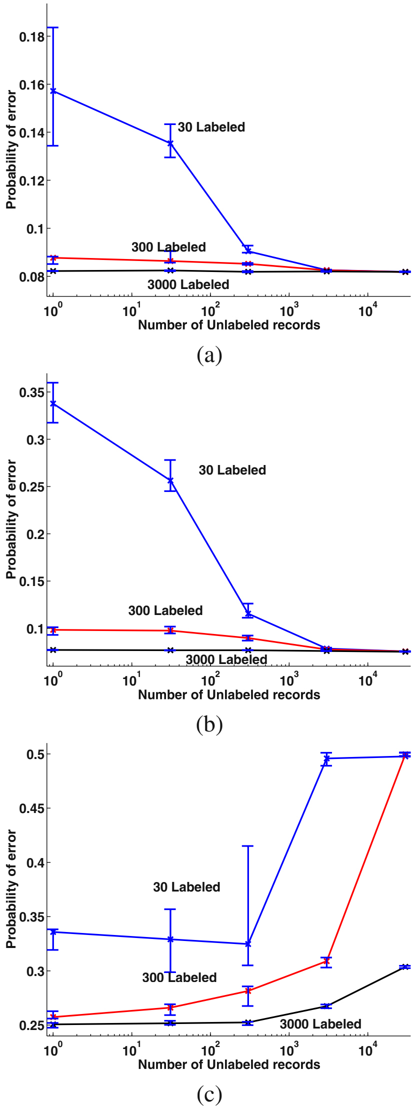
Figure 4.6(a-c) shows an example of the probability of error graphs for the three types of tests we performed, where each point in the graph is an average of 10 trials. The graphs correspond to models with 10 features. Figure 4.6(a) corresponds to learning a NB structure when the correct structure is NB. Figure 4.6(b) is the result of estimating a TAN structure when the correct structure is TAN and Figure 4.6(c) is the result of estimating a NB structure when the correct structure is the TAN given in (b).

We see from Figures 4.6(a) and 4.6(b) that unlabeled data help significantly in reducing the classification error. We also see that the error is reduced further as more unlabeled data are added. When more labeled data are added, the improvement gained by using the unlabeled data is smaller. That can be explained by the fact that the classifier learned using only the labeled data is already close to the optimal Bayes error rate.

The graphs in Figure 4.6(c) show that unlabeled data degrade the performance when an incorrect structure is assumed. First we see that adding more labeled data improves the classification even with an incorrect structure assumption. Second we see that as we add more unlabeled data, the classification error becomes higher, even with a small number of labeled data.

6.2 Can Unlabeled Data Help with Incorrect Models? Bias vs. Variance Effects and the Labeled-unlabeled Graphs

Our asymptotic analysis and the experiments presented above suffice to show the importance of modeling assumption when learning with unlabeled data, but how do we then account for the success of other researchers in applications such as text classification [Nigam et al., 2000], image understanding [Baluja, 1998], and others? There are two possibilities. First, it might be that their assumed model was truly the correct model. Alternatively, a more plausible explanation is that of the tradeoff between bias and variance.



(a)

(b)

(c)

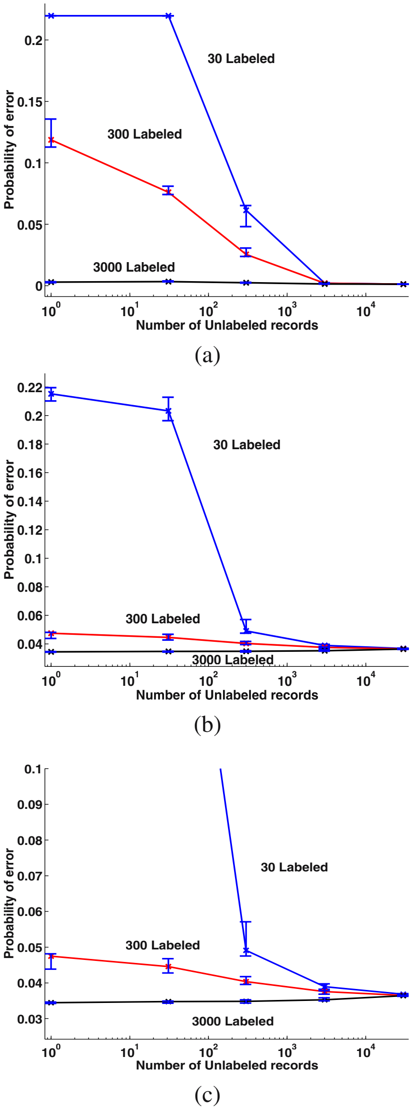
Figure 4.6. Classification error example for networks with 10 features. (a) Assumed and correct structure is NB, (b) Assumed and correct structure is TAN, (c) Assumed structure is NB, correct structure is TAN. The bars represent 30% and 70% percentiles of the error (statistics computed over 10 trials per point).

If we take the application of classifying face orientation [Baluja, 1998], the problem involves many observables (image pixels) with a small corpus of labeled data. From our theoretical analysis we know that regardless of modeling assumptions, the addition of unlabeled data decreases the variance of the estimator, while when the model is incorrect, the estimation bias can increase. Classification error with finite training data is a function of both the bias and the variance [Friedman, 1997; James, 2003]. Therefore, when the amount of labeled data is small, the increase in bias caused by the unlabeled data is mitigated by the decrease in variance, hence causing an improvement in classification performance. This agrees with the conclusions of Shahshahani and Landgrebe [Shahshahani and Landgrebe, 1994a] who indicated that unlabeled data become more useful as the number of observables increases.

To illustrate this point we performed another test, using data generated from a TAN structure with 49 features. The data sets were generated just as in the previous tests. Figure 4.7(a-b) shows the averaged classification error for both types of experiments, with (a) showing the results assuming the correct structure and (b) the results assuming a NB model. Again, we see that when we assume the correct structure, adding the unlabeled examples improves the classification result at a fast rate, reaching almost the Bayes rate with just 30 labeled records and 30000 unlabeled records. In Figure 4.7(b) we see that although the structure assumption is incorrect, adding unlabeled data improves the classification results significantly for the cases where 30 and 300 labeled records were used. However, as can be seen in Figure 4.7(c), with 3000 labeled records, adding unlabeled data degraded the performance. We can conclude that when the estimator using only the labeled data has low variance, adding the unlabeled data can degrade the performance. This means that unlabeled data improve or degrade the classifier's performance depending on both the classifier's complexity and the number of labeled training records.

Further strengthening the experiment above, we performed a similar experiment with the Adult database taken from the UCI repository. The Adult database consists of 30162 labeled records for training and 15060 labeled records for testing. The classification problem in the Adult database is to correctly estimate the income of a person (above or below $50K per year) using 15 features, such as age, sex, profession, etc. The dataset is the result of the U.S. Census bureau questioners. The study by Kohavi [Kohavi, 1996] using the MLC++ machine learning library showed that the classification error using all of the labeled data set is between 14-17% for the best classifiers. Naive-Bayes was used as one of the classifiers, and it achieved around 16% classification error.

In our experiment, we randomly partition the training data set to create labeled and unlabeled (LUL) data sets; ranging from 30–3000 for the labeled sets and 0–30000 for the unlabeled sets. When possible, we create 5 sample



(a)

(b)

(c)

Figure 4.7. Classification error example for networks with 49 features. (a) Assumed and correct structure are TAN, (b) Assumed structure is NB, correct structure is TAN, (c) Zoom in on the bottom portion of (b).The bars represent 30% and 70% percentiles of the error (statistics computed over 10 trials per point).

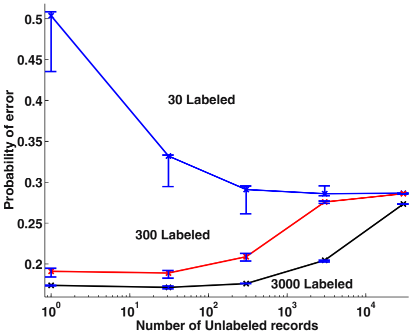


Figure 4.8. Classification result for the Adult DB experiment. The bars represent 30% and 70% percentiles of the error (statistics computed over the five trials per point).

sets. We use the EM algorithm to learn a NB classifier for each LUL training set.

The classification results are shown in Figure 4.8. The graphs clearly show that using unlabeled data increases the classification error compared to using only labeled data, except for the cases with only 30 labeled records.

Similarly to the artificially generated data case of Figure 4.7, when only 30 labeled records are available adding the unlabeled data does improve the classification result, from about 50% error to 30% error. However, as the size of the labeled set increases, adding unlabeled data degrades the classification performance from 19% and 17% to about 30%. This result also indicates that the underlying structure is not NB.

To visualize the effect of labeled and unlabeled samples, going from a small dataset to a large dataset, we suggest fixing the *percentage* of unlabeled samples (λ) among all training samples, and then plotting classification error against the number of training samples. We call such a graph a *labeled-unlabeled (LU) graph.*

EXAMPLE 4.8 *Consider a situation where we have a binary class variable C with values c' and c'', and $p(C = c') = 0.4017$. We also have two real-valued observables X and Y with distributions:*

$$p\big(X|c'\big) = N(2,1), \quad p\big(X|c''\big) = N(3,1),$$

$$p\big(Y|c', x\big) = N(2,1), \quad p\big(Y|c'', x\big) = N(1+2x,1).$$

There is dependency between Y and X conditional on $\{C = c''\}$. Suppose we build a Naive Bayes classifier for this problem. Figure 4.9(a) shows LU-graphs for 0% unlabeled samples, 50% unlabeled samples and 99% unlabeled samples, averaging over a large ensemble of classifiers. As expected, the asymptotes converge to different values. Suppose then that we started with 50 labeled *samples as our training data. Our classification error would be about 7.8%, as we can see in the LU-graph for 0% unlabeled data. Suppose we added 50 labeled samples; we would obtain a classification error of about 7.2%. Now suppose we added 100* unlabeled *samples. We would move from the 0% LU-graph to the 50% LU-graph. Classification error would increase to 8.2%! And if we then added 9800 unlabeled samples, we would move to the 99% LU-graph, with classification error about 16.5% — more than twice the error we had with just 50 labeled samples.* □

Similar to Example 4.8, Figure 4.9(b) shows the LU graphs for a case where the features are discrete, involving 10 features sampled from a TAN model and learned with a NB model. As expected, the LU-graphs display similar behavior as with the mixture of Gaussian case.

In difficult classification problems, where LU-graphs decrease very slowly, unlabeled data may improve classification performance. As we saw in the Figures 4.8 and 4.7, problems with a large number of observables and parameters should require more training data, so we can expect that such problems benefit more consistently from unlabeled data. Another possible phenomenon is that the addition of a substantial number of unlabeled samples may reduce variance and decrease classification error, but an additional, much larger, pool of unlabeled data can eventually add enough bias so as to increase classification error. Such a situation is likely to have happened in some of the results reported by Nigam et al. [Nigam et al., 2000], where classification errors go up and down as more unlabeled samples are added.

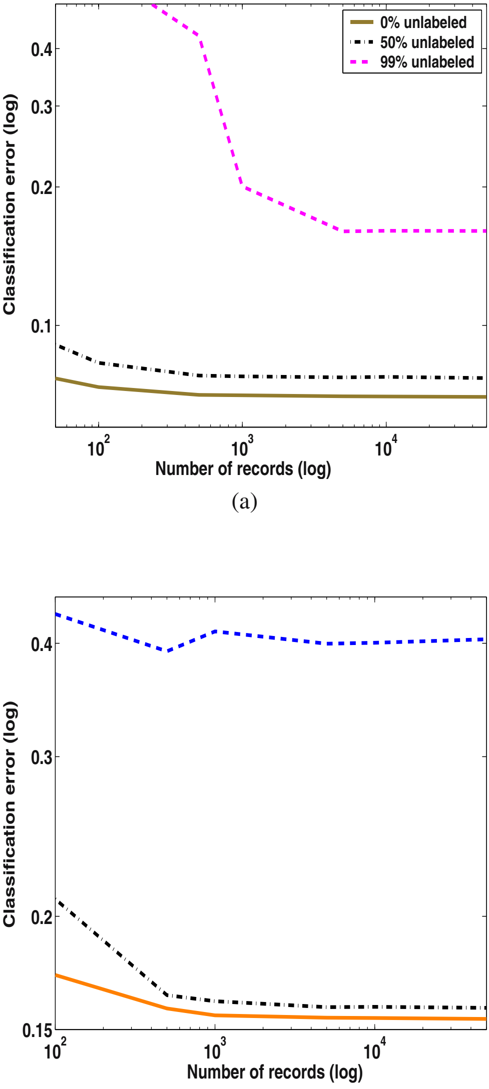
6.3 Detecting When Unlabeled Data Do Not Change the Estimates

From all the results discussed so far, we should expect the unlabeled data to affect the estimates of the classifier's parameters (improving or degrading classification). But, can we predict those situations where unlabeled data do not affect the initial estimate? Note that this question focuses on a specific data set, not on expected behavior. The remainder of this section discusses this issue.

Given that all of the observables are discrete, the empirical distribution for $\mathbf{X}$ in the unlabeled data is:

$$f_u(\mathbf{X} = \mathbf{x}) = \frac{\#\text{ of times } \{\mathbf{X} = \mathbf{x}\} \text{ in unlabeled records}}{N_u}.$$

Given a particular dataset D and a given joint probability distribution $p(C, \mathbf{X}|\theta)$, we prove the following theorem.



(a)

(b)

Figure 4.9. LU-graphs (a) example with two Gaussian features, (b) Discrete features. Each point in each graph is the average of multiple trials.

THEOREM 4.9 *Assume that all the features in the vector X are discrete and there exists a θ such that $p(\mathbf{X}|\theta) = f_u(\mathbf{X})$. Let $\theta_l^* = \arg\max_\theta L_l(\theta)$ and $\theta^* = \arg\max_\theta L(\theta)$. If for all values of X, $p(\mathbf{X}; \theta_l^*) = f_u(\mathbf{X})$, then $\theta^* = \theta_l^*$.*

Proof. We know that if for some θ', $p(\mathbf{X}|\theta') = f_u(\mathbf{X})$, then $\theta' = \arg\max_\theta L_u(\theta)$.

So for $\theta_l^* = f_u(\mathbf{x})$ we obtain $\theta_l^* = \arg\max_\theta L_u(\theta)$.

Now we can bound $\max L(\theta)$ from above using:

$$\begin{aligned}\max L(\theta) &= \max L_l(\theta) L_u(\theta) \\ &\leq (\max L_l(\theta)) (\max L_u(\theta)) \\ &= L_l(\theta_l^*) L_u(\theta_l^*),\end{aligned}$$

and then clearly the way to maximize $L(\theta)$ is to take $\theta = \theta_l^*$.

The theorem states that if the empirical marginal of the unlabeled data is equal to $p(\mathbf{X}|\theta_l^*)$, then the unlabeled data do not change the estimate of the model. Note that it is easy to compute θ_l^* for a given dataset by simple event counting, since the data is fully labeled and we assume no missing data for the features.

When labeled data are available in abundance, then θ_l^* should be enough to provide a good approximation to the empirical marginal, and then the value of unlabeled data is small. It is also important to note that the theorem relates to a specific dataset. It is expected that for small size datasets, the condition that $p(\mathbf{X}|\theta_l^*) = f_u(\mathbf{X})$ is unlikely to be be met, perhaps strengthening the notion that unlabeled data typically do change the estimates of the distribution.

6.4 Using Unlabeled Data to Detect Incorrect Modeling Assumptions

The analysis of the previous sections presents a new, potentially powerful, use for unlabeled data: detecting incorrect modeling assumptions. Consider the following setup. We have a dataset with sufficient labeled data to estimate classification error (using data partition or cross validation), and a much larger dataset with unlabeled data. We want to obtain estimates $\hat{\theta}$ for a parametric model $p(C, \mathbf{X}|\theta)$ that supposedly contains $p(C, \mathbf{X})$. We can test the validity of the model by examining whether estimates with labeled data and estimates with labeled and unlabeled data produce distinct classification errors. Consider the null hypothesis that the parametric model $p(C, \mathbf{X}|\theta)$ in fact contains the "true" distribution $p(C, \mathbf{X})$. We can use results by O'Neill [O'Neill, 1978] to obtain the asymptotic distribution of classification error under this null hypothesis. Alternatively, a bootstrap scheme can produce the distribution of classification error with labeled samples and with labeled and unlabeled samples [Efron and Tibshirani, 1993]. We then test whether the classification error obtained with unlabeled data is identical to the classification error obtained with labeled data, thus validating the null hypothesis.

In Chapter 7, we will use such a test to decide on when to switch between a small set of models (Naive Bayes and TAN). However, this test can be used without having to switch between models, and whenever data are abundant, the test is a strong indicator of model validity.

7. Concluding Remarks

In this chapter, we have demonstrated the different properties of learning classifiers with unlabeled data. We showed that the optimistic view on the value of unlabeled data, while valid with correct modeling assumptions, is often wrong with incorrect models. The asymptotic analysis of the maximum likelihood estimators, followed by the different examples, prove that unlabeled data can have a deleterious effect on the classifier's performance. Experiments with finite data sets displayed the same phenomenon whenever the increase of bias is more significant than the decrease in variance, both attributed to the addition of unlabeled data.

An obvious conclusion of the analysis is that finding models that are closer to the true data generating distribution is very important in semi-supervised learning, perhaps much more than in the purely supervised case. In Chapter 7, we discuss this conclusion in the context of Bayesian network classifiers.

Following our investigation of semi-supervised learning, there are several important open theoretical questions and research directions:

- Is it possible to find necessary and sufficient conditions for performance degradation to occur? Finding such conditions are of great practical significance. Knowing these conditions can lead to the design of new useful tests that will indicate when unlabeled can be used or when they should be discarded, or a different model should be chosen?

- An important question is whether other semi-supervised learning methods, such as transductive SVM [Bennett and Demiriz, 1998], co-training [Blum and Mitchell, 1998] will exhibit the phenomenon of performance degradation? While no extensive studies have been performed, a few results from the literature suggest that it is a realistic conjecture. Zhang and Oles [Zhang and Oles, 2000] demonstrated that transductive SVM can cause degradation of performance when unlabeled data are added. Ghani [Ghani, 2002] described experiments where the same phenomenon occurred with co-training. If the causes of performance degradation are the similar for different algorithms, it should be possible to present a unified theory for semi-supervised learning.

- Are there performance guarantees for semi-supervised learning with finite amounts of data, labeled and unlabeled? In supervised learning such guarantees are studied extensively. PAC and risk minimization bounds help in determining the minimum amount of (labeled) data necessary to learn a classifier with good generalization performance. However, there are no existing bounds on the classification performance when training with labeled and unlabeled data. Finding such bounds can be derived using principals in estimation theory, based on asymptotic covariance properties of the estimator. Other bounds can be derived using PAC theoretical approaches. Existence of such bounds can immediately lead to new algorithms and approaches, better utilizing unlabeled data.

- Can we use the fact that unlabeled data indicates model incorrectness to actively learn better models? The use of active learning seems promising whenever possible, and it might be possible to extend active learning to learn better models, not just enhancement of the parameter estimation.

Chapter 5

ALGORITHM: MAXIMUM LIKELIHOOD MINIMUM ENTROPY HMM

In Chapter 2, we analyzed the probabilistic classifiers and observed that when probability distributions are used for classification task, the classification performance is very closely related to the cross entropy between the hidden and the observed states. In this chapter, we use this to develop a new learning algorithm for the HMM.

We formalize the idea of using information theory in the framework of Hidden Markov Models (HMMs). In the case of HMMs, we enforce the hidden state variables to capture relevant information about the observations. At the same time, we would like our models to explain the generative process of the data as accurately as possible. Therefore, we propose a cost function that combines both the information theoretic (MI) and the maximum likelihood (ML) criteria.

In later chapters (Chapters 8 and 9), we will see that HMMs are quite successful in modeling the temporal data and good performance is achieved when HMMs are used as classifiers.

1. Previous Work

Numerous variations of the standard formulation of Hidden Markov Models have been proposed in the past, such as Parameterized-HMM (PHMM) [Wilson and Bobick, 1998], Entropic-HMM [Brand and Kettnaker, 2000], Variable-length HMM (VHMM) [Galata et al., 2001], Coupled-HMM (CHMM) [Brand et al., 1997], Input-Output HMM (IOHMM) [Bengio and Frasconi, 1996], Factorial-HMM [Ghahramani and Jordan, 1996], and Hidden-Markov Decision Trees (HMDT) [Jordan et al., 1997], to name a few. Each of these models attempts to solve some of the deficiencies of standard HMMs given the particular problem or set of problems at hand. Given that most of them aim at modeling the data and learning the parameters using ML, in many cases their main differences lie in the conditional independence assumptions made while modeling the data, i.e. in their graphical structure. Conversely, the graphical structure of the model presented in this chapter remains the same as that of a standard HMM, but the optimization function is different. Even though we develop here the learning equations for HMMs, the framework that we present could easily be extended to any graphical model.

Recently, Tishby et al. [Tishby et al., 1999] proposed Information Bottleneck as a method for doing clustering. The Information Bottleneck method is an unsupervised non-parametric data organization technique. Given a joint distribution $P(A, B)$, the method constructs, using information theoretic principles, a new variable T that extracts partitions, or clusters, over the values of A that are informative about B. In particular, consider two random variables X and Q with their assumed joint distribution $P(X, Q)$, where X is the variable that we are trying to compress with respect to the 'relevant' variable Q. Tishby et al. [Tishby et al., 1999] propose the introduction a soft partitioning of X through an auxiliary variable T, and the probabilistic mapping $P(T|X)$, such that the mutual information $I(T; X)$ is minimized (maximum compression) while the probabilistic mapping $P(T|X)$, the relevant information $I(T; Q)$ is maximized.

This model is also related to the recently popular debate of conditional versus joint density estimation. The 'conditional' approach (i.e. the maximization of the conditional likelihood of the variables of interest instead of the full likelihood) is closely related to the use of discriminative approaches in learning theory. Jebara and Pentland [Jebara and Pentland, 1998] nicely summarize the advantages an sadvantages associated with joint and conditional density estimation. Standard HMMs perform joint density estimation of the hidden state and observation random variables. However, in situations where the resources are limited (complexity, data, structures), the system has to handle very high dimensional spaces or when the goal is to classify or cluster with the learned models, a conditional approach is probably superior to the full joint density approach. One can think of these two methods (conditional vs joint) as two extremes with our work providing a tradeoff between the two. Sections 5.2 and 5.4 analyze the properties of our approach and relate it to the purely probabilistic model more formally.

Finally we would like to point out how our work is different to the Maximum Mutual Information Estimation (MMIE) approach that is so popular in the speech recognition community. In particular, Bahl et al. [Bahl et al., 1993] introduced the concept of Maximum Mutual Information Estimation (MMIE) for estimating the parameters of an HMM in the context of speech recognition, where typically a different HMM is learned for each possible class (e.g. one HMM for each word in the vocabulary). New waveforms are classified by computing their likelihood based on each of the models. The model with the highest likelihood is selected as the winner. However, in our approach, we learn a single HMM whose hidden states correspond to different classes. The algorithm in [Bahl et al., 1993] attempts to maximize the mutual information between the choice of the HMM and the observation sequence to improve the discrimination across different models. In contrast, our algorithm aims at maximizing the mutual information between the observations and the hidden states, so as to minimize the classification error when the hidden states are used as the classification output.

2. Mutual Information, Bayes Optimal Error, Entropy, and Conditional Probability

In the 'generative approach' to machine learning, the goal is to learn a probability distribution that defines the process that generated the data. Such an approach is particularly good in modeling the general form of the data and can give some useful insights into the nature of the original problem. Recently, there has been an increasing focus on connecting the performance of these generative models to their classification accuracy when they are used for classification tasks. Recall that in Chapter 2 we develop an extensive analysis of the relationship between the Bayes optimal error of a classification task using a probability distribution and the entropy between the random variables of interest. Consider the family of probability distributions over two random variables (X, Q) denoted by $P(X, Q)$. The classification task is to predict Q after observing X. As given in Theorem 2.4, the relationship between the conditional entropy $H(X|Q)$ and the Bayes optimal error, ϵ is given by

$$\frac{1}{2}H_b(2\epsilon) \leq H(X|Q) \leq H_b(\epsilon) + \log\frac{N}{2} \tag{5.1}$$

with $H_b(p) = -(1-p)\log(1-p) - p\log p$.

Figure 5.1 illustrates this relationship between the conditional entropy and the Bayes optimal error. In Figure 5.1 the only realizable –and at the same time observable– distributions are those within the black region. One can conclude from Figure 5.1 that, if the data is generated according to a distribution that has high conditional entropy, the Bayes optimal error of any classifier for this data will be high. Even though this relationship is between the *true model* and the *Bayes optimal error*, it also applies to a model that has been estimated from data, – assuming a consistent estimator has been used, such as Maximum Likelihood, and the model structure is the true one. As a result, when the learned distribution has high conditional entropy, it might not necessarily do well on classification. Therefore, if the final goal is classification, the graph in Figure 5.1 suggests that low entropy models should be preferred over high entropy ones. The cost function proposed in Eqn 5.2 favors low conditional entropy models to high entropy ones.

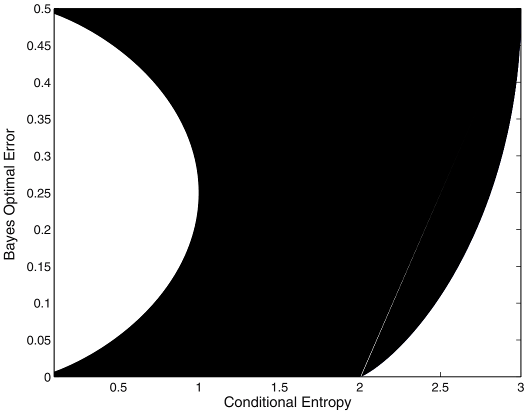


Figure 5.1. Bayes optimal error versus conditional entropy

A Hidden Markov Model (HMM) is a probability distribution over a set of random variables, some of which are referred to as the hidden states (as they are normally not observed and they are discrete) and others are referred to as the observations (continuous or discrete). Traditionally, the parameters of Hidden Markov Models are estimated by maximizing the joint likelihood of the hidden states Q and the observations X, $P(X, Q)$. Conventional Maximum Likelihood (ML) techniques would be optimal in the case of very large datasets (so that the estimate of the parameters is correct) if the true distribution of the data was in fact an HMM. However none of the previous conditions is normally true in practice. The HMM assumption might be in many occasions highly unrealistic and the available data for training is normally very limited, leading to important problems associated with the ML criterion (such as overfitting). Moreover, ML estimated models are often used for clustering or classification. In these cases, the evaluation function is different to the optimization function, which suggests the need of an optimization function that correctly models the problem at hand. The cost function defined in Eqn 5.2 is designed to tackle some of these problems associated to ML estimation.

When formulating our optimization functional, we exploit the relationship between the conditional entropy of the data and the Bayes optimal error previously described. In the case of Hidden Markov Models (HMMs), the X variable corresponds to the observations and the Q variable to the hidden states. We would like to maximize the joint probability distribution $P(Q, X)$ while forcing the Q variable to contain maximum information about the X variable (i.e. to maximize their mutual information or minimize the conditional entropy). In consequence, we propose to maximize both the joint likelihood and the mutual information between the hidden variables and the observations.

This leads to the following cost function

$$F = (1-\alpha)I(Q,X) + \alpha \log P(X_{obs}, Q_{obs}). \tag{5.2}$$

where $\alpha \in [0,1]$, provides a way of deciding the appropriate weighting between the Maximum Likelihood (ML) (when $\alpha = 1$) and Maximum Mutual Information (MMI) (when $\alpha = 0$) criteria, and $I(Q,X)$ refers to the mutual information between the states and the observations. However, very often one does not observe the state sequence[1]. In such a scenario, the cost function reduces to

$$F = (1-\alpha)I(Q,X) + \alpha \log P(X_{obs}). \tag{5.3}$$

3. Maximum Mutual Information HMMs

We develop in this section the learning algorithms for discrete and continuous, supervised and unsupervised Maximum Mutual Information HMMs (MMIHMMs). For the sake of clarity and simplicity, we will start with the supervised case, where the 'hidden' states are actually observed in the training data.

Consider a Hidden Markov Model with **Q** as the states and **X** as the observations. Let F denote the cost function to maximize,

$$F = (1-\alpha)I(Q,X) + \alpha \log P(X_{obs}, Q_{obs}) \tag{5.4}$$

The mutual information term $I(Q,X)$ can be expressed as $I(Q,X) = H(X) - H(X/Q)$, where $H(\cdot)$ refers to the entropy. Since $H(X)$ is independent of the choice of the model and is characteristic of the generative process, we can reduce our cost function to

$$\begin{aligned} F &= -(1-\alpha)H(X/Q) + \alpha \log P(X_{obs}, Q_{obs}) \\ &= (1-\alpha)F_1 + \alpha F_2. \end{aligned}$$

In the following we will use the standard HMM notation for the transition a_{ij} and observation b_{ij} probabilities,

$$a_{ij} = P(q_t = i, q_{t+1} = j), \quad b_{ij} = P(x_t = j | q_t = i). \tag{5.5}$$

Expanding each of the terms F_1 and F_2 separately we obtain,

$$\begin{aligned} F_1 &= -H(X|Q) = \sum_X \sum_Q P(X,Q) \log \prod_{t=1}^{T} P(x_t|q_t) \\ &= \sum_{t=1}^{T} \sum_{j=1}^{M} \sum_{i=1}^{N} P(x_t = j|q_t = i) P(q_t = i) \log P(x_t = j|q_t = i) \\ &= \sum_{t=1}^{T} \sum_{j=1}^{M} \sum_{i=1}^{N} P(q_t = i) b_{ij} \log b_{ij}. \end{aligned}$$

[1] We will refer to this case as the unsupervised case while referring to the former as the supervised case.

and,

$$F_2 = \log \pi_{q_1^o} + \sum_{t=2}^{T} \log a_{q_{t-1}^o, q_t^o} + \sum_{t=1}^{T} \log b_{q_t^o, x_t^o}.$$

Combining F_1 and F_2 and adding the appropriate Lagrange multipliers to ensure that the a_{ij} and b_{ij} coefficients sum to 1, we obtain:

$$\begin{aligned} F_L &= (1-\alpha)\sum_{t=1}^{T}\sum_{j=1}^{M}\sum_{i=1}^{N} P(q_t = i) b_{ij} \log b_{ij} \\ & \quad + \alpha \log \pi_{q_1^o} + \alpha \sum_{t=2}^{T} \log a_{q_{t-1}^o, q_t^o} + \alpha \sum_{t=1}^{T} \log b_{q_t^o, x_t^o} \\ & \quad + \beta_i \left(\sum_j a_{ij} - 1 \right) + \gamma_i \left(\sum_j b_{ij} - 1 \right). \end{aligned} \tag{5.6}$$

Note that in the case of continuous observation HMMs, we can no longer use the concept of entropy as previously defined. As a result, we will be using the counterpart differential entropy. Because of this important distinction, we will carry out the analysis for discrete and continuous observation HMMs separately.

3.1 Discrete Maximum Mutual Information HMMs

To obtain the parameters that maximize the cost function, we take the derivative of F_L from Eqn 5.5 and will equate it to zero. First solving for b_{ij}, we obtain:

$$\begin{aligned} \frac{\partial F_L}{\partial b_{ij}} &= (1-\alpha)(1+\log b_{ij}) \left(\sum_{t=1}^{T} P(q_t = i) \right) + \frac{N_{ij}^b \alpha}{b_{ij}} + \gamma_i \\ &= 0 \end{aligned} \tag{5.7}$$

where N_{ij}^b is the number of times one observes state j when the hidden state is i. Eqn 5.6 can be expressed as

$$\log b_{ij} + \frac{W_{ij}}{b_{ij}} + g_i + 1 = 0 \tag{5.8}$$

where

$$W_{ij} = \frac{N^b_{ij}\alpha}{(1-\alpha)\sum\limits_{t=1}^{T} P(q_t = i)}$$

$$g_i = \frac{\gamma_i}{(1-\alpha)\sum\limits_{t=1}^{T} P(q_t = i)}.$$

The solution of Eqn 5.8 is given by

$$b_{ij} = -\frac{W_{ij}}{\mathsf{LambertW}(-\mathsf{W_{ij}e^{1+g_i}})} \tag{5.9}$$

where $\mathsf{LambertW(x)} = \mathsf{y}$ is the solution of the equation $ye^y = x$.

Now we are going to solve for a_{ij}. Let's first look at the derivative of F_1 with respect to a_{lm}.

$$\frac{\partial F_1}{\partial a_{lm}} = \sum_{t=1}^{T}\sum_{j=1}^{M}\sum_{i=1}^{N} b_{ij}\log b_{ij}\frac{\partial P(q_t = i)}{\partial a_{lm}}. \tag{5.10}$$

To solve the above equation, we need to compute $\frac{\partial P(q_t=i)}{\partial a_{lm}}$. This can be computed using the following iteration:

$$\frac{\partial P(q_t = i)}{\partial a_{lm}} = \begin{cases} \sum\limits_{j} \frac{\partial P(q_{t-1}=j)}{\partial a_{lm}} a_{ji} & \text{if } m \neq i, \\ \sum\limits_{j} \frac{\partial P(q_{t-1}=j)}{\partial a_{lm}} a_{ji} + P(q_{t-1} = l) & \text{if } m = i \end{cases} \tag{5.11}$$

with the initial conditions

$$\frac{\partial P(q_2 = i)}{\partial a_{lm}} = \begin{cases} 0 & \text{if } m \neq i, \\ \pi_l & \text{if } m = i. \end{cases} \tag{5.12}$$

Taking the derivative of F_L, with respect to a_{lm}, we obtain,

$$\frac{\partial F}{\partial a_{lm}} = (1-\alpha)\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{k=1}^{M} b_{ik}\log b_{ik}\frac{\partial P(x_t = i)}{\partial a_{lm}} + \alpha\frac{N_{lm}}{a_{lm}} + \beta_l$$

where N_{lm} is the count of the number of occurrences of $q_{t-1} = l, q_t = m$ in the data set. The update equation for a_{lm} is obtained by equating this quantity to zero and solving for a_{lm}

$$a_{lm} = \frac{\alpha N_{lm}}{(1-\alpha)\sum\limits_{t=1}^{T}\sum\limits_{i=1}^{N}\sum\limits_{k=1}^{M} b_{ik}\log b_{ik}\frac{\partial P(x_t=i)}{\partial a_{lm}} + \beta_l} \tag{5.13}$$

where β_l is chosen so that $\sum_m a_{lm} = 1, \forall l$.

3.2 Continuous Maximum Mutual Information HMMs

For the sake of clarity, we will restrict our attention to the case when the $P(x|q)$ is a single Gaussian. Under this assumption, the HMM is characterized by the following parameters

$$\begin{aligned} P(q_t = j|q_{t-1} = i) &= a_{ij} \\ P(x_t|q_t = i) &= \frac{1}{\sqrt{2\pi|\Sigma_i|}} \exp\left(-\frac{1}{2}(x_t - \mu_i)^T \Sigma_i^{-1}(x_t - \mu_i)\right) \end{aligned}$$

where Σ_i is the covariance matrix when the hidden state is i and $|\Sigma_i|$ is the determinant of the covariance matrix. Now, for the cost function given in Eqn 5.2, F_1 and F_2 can be written as

$$\begin{aligned} F_1 &= -H(X|Q) \\ &= \sum_{t=1}^{T}\sum_{i=1}^{N} \int P(q_t = i) \log P(x_t|q_t = i) dP(x_t|q_t = i) \\ &= \sum_{t=1}^{T}\sum_{i=1}^{N} P(q_t = i) \int \left(-\frac{1}{2}\log(2\pi|\Sigma_i|)\right. \\ &\quad \left. -\frac{1}{2}(x_t - \mu_i)^T \Sigma_i^{-1}(x_t - \mu_i)\right) dP(x_t|q_t = i)y \\ &= \sum_{t=1}^{T}\sum_{i=1}^{N} P(q_t = i)\left(-\frac{1}{2}\log(2\pi|\Sigma_i|) - \frac{1}{2}\right), \end{aligned}$$

and,

$$\begin{aligned} F_2 &= \log P(Q_{obs}, X_{obs}) \\ &= \sum_{t=1}^{T} \log P(x_t|q_t) + \log \pi_{q_1^o} + \sum_{t=2}^{T} \log a_{q_{t-1}^o, q_t^o}. \end{aligned}$$

Following the same steps as for the discrete case, we again form the Lagrange F_L, take its derivative with respect to each of the unknown parameters, and obtain the corresponding update equations. First consider the means of the Gaussian:

$$\mu_i = \frac{\sum\limits_{t=1, q_t=i}^{T} x_t}{N_i} \tag{5.14}$$

where N_i is the number of times $q_t = i$ in the observed data. Note that this is the standard update equation for the mean of a Gaussian, and it is the same as for ML estimation in HMMs. This is because the conditional entropy is independent of the mean.

Next, the update equation for a_{lm} is same as in Eqn 5.13 except for replacing $\sum_k b_{ik} \log b_{ik}$ by $-\frac{1}{2}\log(2\pi|\Sigma_i|) - \frac{1}{2}$. Finally, the update equation for Σ_i is

$$\begin{aligned}\Sigma_i &= \frac{\alpha \sum\limits_{t=1,q_t=i}^{T} (x_t - \mu_i)(x_t - \mu_i)^T}{N_i\alpha + (1-\alpha)\sum\limits_{t=1}^{T} P(q_t = i)} \\ &= \frac{\sum\limits_{t=1,q_t=i}^{T} (x_t - \mu_i)(x_t - \mu_i)^T}{N_i + \frac{(1-\alpha)}{\alpha}\sum\limits_{t=1}^{T} P(q_t = i)}. \qquad (5.15)\end{aligned}$$

It is interesting to note that the update equation for Σ_i in Eqn 5.14 is very similar to the one obtained when using ML estimation, except for the term in the denominator $\frac{(1-\alpha)}{\alpha}\sum_{t=1}^{T} P(q_t = i)$, which can be thought of as a regularization term. Because of this positive term, the covariance Σ_i is smaller than what it would have been otherwise. This corresponds to lower conditional entropy, as desired.

3.3 Unsupervised Case

The above analysis can easily be extended to the unsupervised case, i.e. when only X_{obs} is given and Q_{obs} is not available. In this case, we use the cost function given in Eqn 5.3. The update equations for the parameters are very similar to the ones obtained in the supervised case. The only difference is that now we replace N_{ij} in Eqn 5.6 by $\sum_{t=1,x_t=j}^{T} P(q_t = i|X_{obs})$, N_{lm} is replaced in Eqn 5.13 by $\sum_{t=2}^{T} P(q_{t-1=l}, q_t = m|X_{obs})$, and N_i is replaced in Eqn 5.14 by $\sum_{t=1}^{T} P(q_t = i|X_{obs})$. These quantities can be easily computed using the Baum-Welch algorithm by means of the forward and backward variables.

4. Discussion

4.1 Convexity

From the law of large numbers, it is known that, in the limit (i.e. as the number of samples approaches infinity), the likelihood of the data tends to the negative of the entropy, $P(X) \approx -H(X)$.

Therefore, in the limit, the negative of our cost function for the supervised case can be expressed as

$$\begin{aligned} -F &= (1-\alpha)H(X|Q) + \alpha H(X,Q) \\ &= H(X|Q) + \alpha H(Q). \end{aligned} \tag{5.16}$$

Note that $H(X|Q)$ is a strictly concave function of $P(X|Q)$, and $H(X|Q)$ is a linear function of $P(Q)$. Consequently, in the limit, the cost function from Eqn 5.15 is strictly convex (its negative is concave) with respect to the distributions of interest.

In the unsupervised case and in the limit again, our cost function can be expressed as

$$\begin{aligned} F &= -(1-\alpha)H(X|Q) - \alpha H(X) \\ &= -H(X) + (1-\alpha)(H(X) - H(X|Q)) \\ &= -H(X) + (1-\alpha)I(X,Q) \approx P(X) + (1-\alpha)I(X,Q). \end{aligned}$$

The unsupervised case thus reduces to the original case with α replaced by $1-\alpha$. Maximizing F is, in the limit, the same as maximizing the likelihood of the data and the mutual information between the hidden and the observed states, as expected.

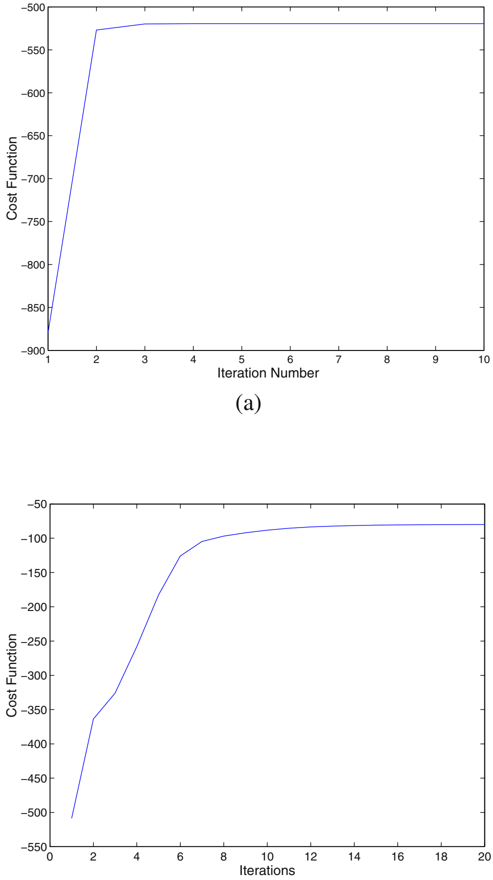
4.2 Convergence

We analyze next the convergence of the MMIHMM learning algorithm in the supervised and unsupervised cases. In the supervised case, HMMs are directly learned without any iteration. However, in the case of MMIHMM we do not have a closed form solution for the parameters b_{ij} and a_{ij}. Moreover these parameters are inter-dependent (i.e. in order to compute bij, we need to compute $P(q_t = i)$ which requires the knowledge of a_{ij}). Therefore an iterative solution is needed. Fortunately, the convergence of the iterative algorithm is extremely fast, as it is illustrated in Figure 5.2. This figure shows the cost function with respect to the iterations for a particular case of the speaker detection problem (a) (see section 5.5.2), and for synthetically generated data in an unsupervised situation (b). From Figure 5.2 it can be seen that the algorithm typically converges after only 2-3 iterations.

4.3 Maximum A-posteriori View of Maximum Mutual Information HMMs

The MMIHMM algorithm presented can also be viewed as a maximum a-posteriori (MAP) algorithm with the conditional entropy acting as a prior on the space of models.

An HMM is a probability distribution over a set of RV's, (X, Q). Traditionally, the parameters of HMMs are estimated by maximizing the joint likelihood



(a)

(b)

Figure 5.2. Value of the cost function with respect to the iteration number in (a) the speaker detection experiment; (b) a continuous unsupervised case with synthetic data.

of the hidden states and the observations, $P(X, Q)$. The work in this chapter can be thought of as an entropic estimation (similar to [Brand, 1998]) framework in the space of possible distributions modeled by an HMM. In contrast to Brand et al. [Brand, 1998], where the prior is imposed over the parameters of the model, we impose the priors directly on the model, preferring models with low conditional entropy. In particular, given a Hidden Markov Model χ, characterized by its parameters $\{\pi, A, B\}$, where π are the initial state probabilities, A is the transition probability matrix, and B is the observation probability matrix (in the discrete case), the prior probability of the model is assumed to be $P(\chi) \propto e^{\lambda I(X;Q)}$. Under this prior, the posterior probability can be written as:

$$P_{post} \equiv P(X,Q|\chi)P(\chi) \;\; \propto \;\; P(X,Q|\chi)e^{\lambda I(X,Q)}$$
$$= \;\; P(X,Q|\chi)e^{\lambda(H(X)-H(X|Q))}. \quad (5.17)$$

The prior $P(\chi) \propto e^{\lambda I(X,Q)}$ is referred to as the entropic prior (modulo a normalization constant) over the space of distributions, preferring distributions with high mutual information over distributions with low mutual information. The parameter λ controls the weight of the prior and acts as a smoothing factor: if λ is very small, all the models are almost equally likely, whereas if λ is large, models with high mutual information are favored over others. Our goal with the HMMs is to predict the hidden states based on the observations. Thus, the mutual information is used to model the dependence between the hidden and the observed states. This concept of entropic priors could be extended to other graphical models, by computing the mutual information between the observed and the query variables. Note how the prior is over the possible distributions and not over the parameters, as proposed in the past [Brand, 1998]. Given that the dependence of $H(X)$ on the model parameters is weak, we will approximate (to keep the problem tractable) the objective function with, $P_{post}(\chi) \propto P(X,Q|\chi)e^{-\lambda H(X|Q)}$. The prior distribution ($e^{-\lambda H(X|Q)}$) can now be seen as favoring the distributions with low conditional entropy.

This prior has two properties derived from the definition of entropy:

1 It is a bias for compact distributions having less ambiguity, *i.e.* lower conditional entropy;

2 It is invariant to re-parameterization of the model because the entropy is defined in terms of the model's joint and/or factored distributions.

Taking the logarithm of the posterior probability in Eqn 5.16 and dropping from now on the explicit dependence on χ, we obtain:

$$F = \log(P_{post}) \;\; \equiv \;\; \log P(X,Q) + \lambda I(X,Q)$$
$$= \;\; \log P(X,Q) + \lambda(H(X) - H(X|Q)). \quad (5.18)$$

This leads to the following function to maximize:

$$F \;\; = \;\; \lambda I(Q,X) + \log P(X_{obs}, Q_{obs})$$
$$= \;\; (1-\alpha)I(Q,X) + \alpha \log P(X_{obs}, Q_{obs}) \quad (5.19)$$

where α, provides a way of trading off between the ML ($\alpha = 1$) and Maximum Mutual Information (MMI) ($\alpha = 0$) criteria, and $\lambda = \frac{(1-\alpha)}{\alpha}$. Note that Eqn 5.3 is exactly the log-posterior probability expressed in Eqn 5.18.

5. Experimental Results

In this section, we describe the set of experiments that we have carried out to obtain quantitative measures of the performance of MMIHMMs when compared to HMMs in various classification tasks. We have conducted experiments with synthetic and real, discrete and continuous, supervised and unsupervised data.

5.1 Synthetic Discrete Supervised Data

We generated 10 different datasets of randomly sampled synthetic discrete data with 4 hidden states and 5 observation values. We used 100 samples for training and 100 for testing. The training was supervised for both HMMs and MMIHMMs. MMIHMMs had an average improvement over the 10 datasets of $\mathbf{12}\%$, when compared to HMMs of exactly the same structure. The optimal α variables ranged from 0.05 to 0.95, depending on the dataset.The best accuracy of HMMs and MMIHMMs for each of the 10 datasets is depicted in Figure 5.3, together with the optimal α for each of the datasets. A summary of the accuracy of HMMs and MMIHMMs is shown in Table 5.1.

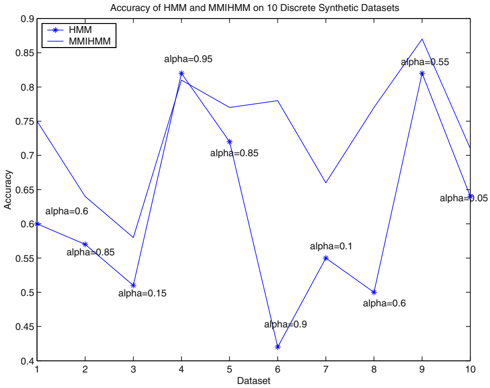


Figure 5.3. Accuracies and optimal value of α for MMIHMM and HMM (star-line) on 10 different datasets of synthetic discrete data.

5.2 Speaker Detection

An estimate of the person's state is important for the reliable functioning of any interface that relies on speech communication. In particular, detecting when users are speaking is a central component of open mike speech-based user interfaces, specially given their need to handle multiple people in noisy environments. We carried out some experiments in a speaker detection task. The speaker detection dataset was the same that appeared in [Garg et al., 2000b]. It consisted of five sequences of one user playing blackjack in a simulated casino setup using CRL's Smart Kiosk [Christian and Avery, 1998]. The sequences were of varying duration from 2000 to 3000 samples, with a total of 12500 frames. The original feature space had 32 dimensions that resulted from quantizing five binary features (skin color presence, face texture presence, mouth motion presence, audio silence presence and contextual information). Only the 14 most significant dimensions were selected out of the original 32-dimensional space.

The learning task in this case was supervised for both HMMs and MMIHMMs. Three were the variables of interest: the presence/absence of a speaker, the presence/absence of a person facing frontally, and the existence/absence of an audio signal or not. The goal was to identify the correct state out of four possible states:

1 no speaker, no frontal, no audio;

2 no speaker, no frontal, and audio;

3 no speaker, frontal, and no audio;

4 speaker, frontal, and audio.

Figure 5.4 illustrates the classification error for HMMs (dotted line) and MMIHMMs (solid line) with α varying from 0.05 to 0.95 in .1 increments. Note how in this case MMIHMMs outperformed HMMs for all the values of α. The accuracies of HMMs and MMIHMMs are summarized in table 5.1. The accuracy reported in [Garg et al., 2000b] using a bi-modal (audio and video) DBN was of about 80%.

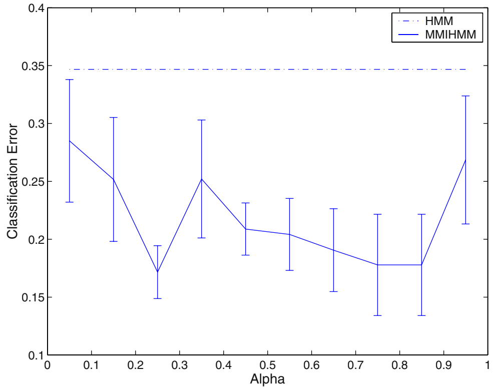


Figure 5.4. Error bars for the Speaker Detection data in MMIHMMs and HMMs

Table 5.1. Classification accuracies for HMMs and MMIHMMs on different datasets

DATASET	HMM	MMIHMM
SYNTHETIC	55%	$\mathbf{66\%}(\alpha_{\mathbf{optimal}} = \mathbf{.1})$
SPEAKERID	64%	$\mathbf{88\%}(\alpha_{\mathbf{optimal}} = \mathbf{.75})$
GENE	68%	$\mathbf{84\%}(\alpha_{\mathbf{optimal}} = \mathbf{.5})$
EMOTION	67%	$\mathbf{74\%}(\alpha_{\mathbf{optimal}} = \mathbf{.8})$

5.3 Protein Data

Gene identification and gene discovery in new genomic sequences is certainly an important computational question addressed by bioinformatics scientists. In this example, we tested both HMMs and MMIHMMs in the analysis of the Adh region in Drosophila. More specifically, part of an annotated drosophila sequence was used to conduct the experiments and to obtain the measure for the algorithms' performance (7000 data points on training and 2000 on testing). MMIHMMs were superior to HMMs for different values of alpha. The best results were obtained for a value of alpha of 0.5 as Table 5.1 reflects.

5.4 Real-time Emotion Data

Finally we carried out an emotion recognition task using the emotion data described in [Cohen et al., 2000]. The data had been obtained from a database of five people that had been instructed to display facial expressions corresponding to the following six types of emotions: anger, disgust, fear, happiness, sadness and surprise. The data collection method is described in detail in [Cohen et al., 2000]. We used the same video database as the one used in [Cohen et al., 2000]. It consisted of six sequences of each facial expression for each of the five subjects. In the experiments reported here, we used unsupervised training of continuous HMMs and MMIHMMs. The accuracy results of both types of models are displayed in Table 5.1.

6. Summary

We have presented a new framework for estimating the parameters of Hidden Markov Models. We have motivated, proposed, and justified a new cost function that linearly combines the mutual information and the likelihood of the hidden states and the observations in an HMM. We have derived the parameter estimation equations in the discrete and continuous, supervised and unsupervised cases. Finally, we have shown the superiority of our approach in a classification task when compared to standard HMMs in different synthetic and real datasets.

Future lines of research include automatically estimating the optimal α, extending the approach to other graphical models with different structures, and better understanding the connection between MMIHMMs and other information theoretic and discriminative approaches. We are also exploring how to apply our framework to a number of applications and real-life problems.

Chapter 6

ALGORITHM: MARGIN DISTRIBUTION OPTIMIZATION

In Chapter 3, we have introduced a new, data-dependent, complexity measure for learning and use it to develop improved generalization bounds. The complexity measure – *projection profile* – is a function of the *margin distribution* – the distribution of the distance of instances from a separating hyperplane.

In this chapter, we show that the projection profile can be used to derive a new learning algorithm that optimizes performance with respect to the generalization error – the Margin Distribution Optimization Algorithm (*MDO*). Experimental results on some real world problems (face detection and context sensitive spelling correction) and some UCI data sets demonstrate the superiority of MDOover Boosting and SVM.

1. Introduction

The study of generalization abilities of learning algorithms and their dependence on sample complexity is one of the fundamental research efforts in learning theory. Understanding the inherent difficulty of learning problems allows one to evaluate the possibility of learning in certain situations, estimate the degree of confidence in the predictions made, and is crucial in understanding, analyzing, and developing improved learning algorithms. In this chapter, we show how the bounds developed in Chapter 3 can be applied towards developing new learning algorithms.

As introduced in Chapter 3, *projection profile* of data sampled according to a distribution $\mathcal{D}$, is the expected amount of error introduced when a classifier h is randomly projected, along with the data, into k-dimensions. It was shown to be captured by the following quantity:

$$a_k(\mathcal{D}, h) = \int_{x \in \mathcal{D}} u(x) d\mathcal{D},$$

where

$$u(x) = \min\left(3\exp\left(-\frac{(\nu(x))^2 k}{8(2+|(\nu(x))|)^2}\right), \frac{2}{k\nu(x)^2}, 1\right) \tag{6.1}$$

and $\nu(x)$ is the distance between x and the classifying hyperplane defined by h, a linear classifier for $\mathcal{D}$. The sequence

$$\mathcal{P}(\mathcal{D}, h) = (a_1(\mathcal{D}, h), a_2(\mathcal{D}, h), \ldots)$$

is the *projection profile* of $\mathcal{D}$. Note that our analysis does not assume linearly separable data.

The projection profile turns out to be quite informative, not just in theory (as was shown in Chapter 3) but even in practice. As we show now, the projection profile of the observed data with respect to a learned classifier can be directly optimized, yielding a new learning algorithm for linear classifiers, MDO (Margin Distribution Optimization), that, as we justify theoretically and show experimentally, outperform existing algorithms for linear functions such as Perceptron, SVM, and boosting. The Margin Distribution Optimization (MDO) algorithm exploits an explicit dependency on the distribution of the geometric distances of points from the classifier – the *margin distribution* of the data – rather than only the extreme points as is done in algorithms like SVM. This is significant when most of the data is far from the optimal classifier - only very few points, those that determine the margin, are close to it. Our experiments reveal that this is indeed the case in many real applications. In these cases, as was shown, our bound is better than existing bounds, and in some cases is informative for very high dimensional data. As a result, MDO outperforms existing algorithms.

In this chapter, we will follow the same notation as in Chapter 3.

2. A Margin Distribution Based Bound

As shown in Chapter 3, for a classifier h and data points x, $\nu(x) = h^T x$ can be thought of as the geometric distance between x and the hyperplane orthogonal to h that passes through the origin (provided $||h|| = ||x|| = 1$). Given a distribution on data points x, this induces a distribution on their distance from the hyperplane induced by h, which is referred to as the *margin distribution*. For a classifier whose decision is based on $sign(h^T x)$, the following theorem was derived in Chapter 3.

THEOREM 6.1 *Let* $S = \{(x_1, y_1), \ldots, (x_{2m}, y_{2m})\}$ *be a set of* n*-dimensional labeled examples and* h *a linear classifier. Then, for all constants* $0 < \delta < 1; 0 < k$, *with probability at least* $1 - 4\delta$, *the expected error of* h *is* *bounded by*

$$\overline{E} \leq \widehat{E}(S,h) + \min_{k} \left\{ \mu_k + 2\sqrt{\frac{(k+1)\ln\frac{me}{k+1} + \ln\frac{1}{\delta}}{2m}} \right\} + \frac{\log m}{m} \tag{6.2}$$

where

$$\mu_k = \frac{2}{m\delta} \sum_{j=1}^{2m} \min\left\{ 3\exp\left(-\frac{\nu_j^2 k}{8(2+|\nu_j|)^2} \right), \frac{2}{\nu_j^2 k}, 1 \right\},$$

and

$$\nu_j = h^T x_j.$$

Informally, it indicates that if "many" of the high dimensional points are classified with high confidence, that is, $|h^T x|$ is large for these, then one does not need as many training examples for good generalization as predicted by VC-theory or margin based theory.

3. Existing Learning Algorithms

In this chapter, we focus our attention on linear learning algorithms. It has been shown that any learning algorithm can be mapped to some linear algorithm in high dimensional space and thus arguing that our analysis is general enough. The aim of any learning algorithm is to learn a classifier that achieves best classification performance on the data sampled according to a certain distribution. However, since only a limited amount data is available during the training phase, intuitively one would expect to choose a classifier that will do the best job on the training data. This intuition is further attested by the Vapnik's result [Vapnik, 1998] on Empirical risk minimization principle.

Some of the popular learning algorithms are Perceptron [Rosenblatt, 1958], Winnow [Littlestone, 1987], and SVM [Vapnik, 1998]. All three algorithms learn a linear classifier. The first two learn in an online fashion whereas the last one is a batch mode learning algorithm. In all three cases learned classifier is a hyperplane and the classification is done by computing the sign of the dot product of the classifier with the data point (that is one which side of the hyperplane where data lies). If the data is noisy then, one may argue that the best classifier is the one with the minimum error on the training set. However, this gets tricky when there are a number of classifier with the same error.

For simplicity, let us consider the case when the training data is linearly separable. Fig. 6.1 shows the training data with "+" corresponding to positive data and "−" corresponding to negative data. The task is to learn a classifier that will be able to classify this data automatically with the same labels. It turns out that in this case there are a number of linear classifiers that can do perfect

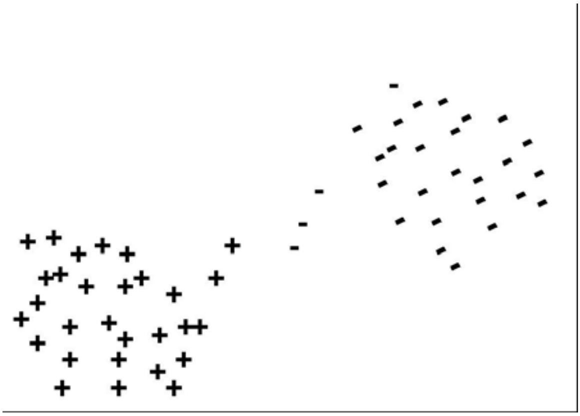
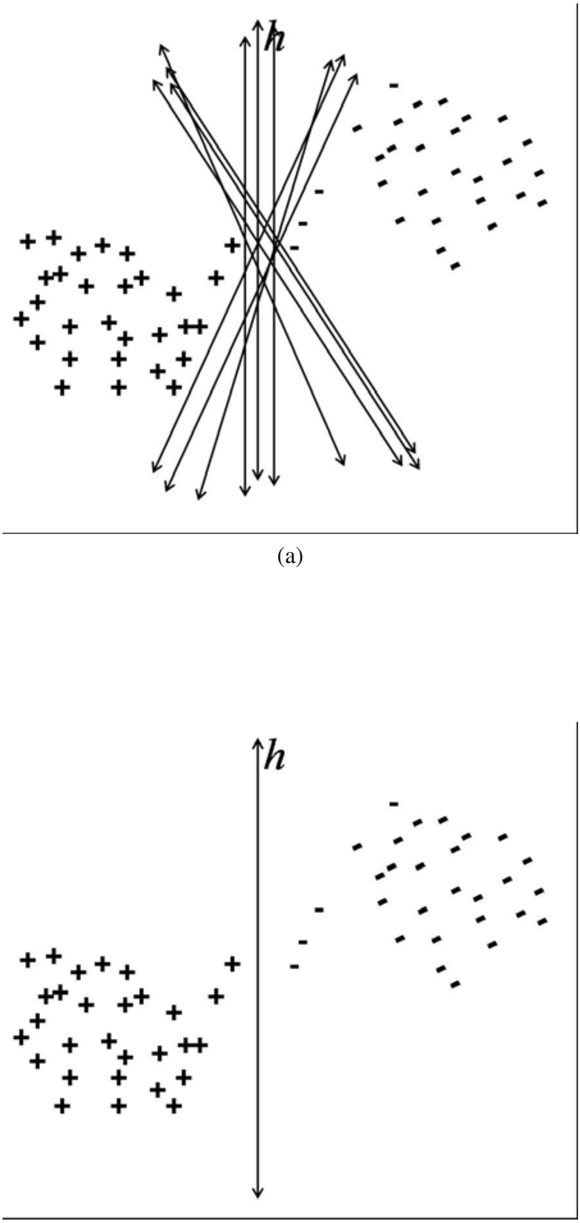


Figure 6.1. A graph showing the training data in two dimensional space with "+" corresponding to positive class and "−" corresponding to negative class.

classification on the training data. Fig. 6.2(a) shows some of such classifiers. Learning algorithms like Perceptron and winnow are mistake driven algorithms that learn in the online fashion. The only criteria that drives the learning is the mistake on the training dataset and the learning stops once a classifier is found that has zero error on the training data. As such depending upon the initialization and the order in which data is presented these algorithms will learn one of the classifier among the ones shown in Fig. 6.2(a) without preferring one over other. Fig. 6.2(b) shows a possible classifier that may be learned by these algorithms.

This is where support vector machine (SVM) distinguishes itself from other algorithms by choosing a classifier that separates the data with the maximum margin. The margin (as defined in Chapter 3) is the distance of the closest point to the learned hyperplane. SVM focuses on only the points closest to the hyperplane and chooses the plane such that it separates the data with the maximum margin. Fig. 6.3(a) shows the learned hyperplane for this data by SVM algorithm. The circled points are the support vectors (the data points closest to the hyperplane) which played the deciding role in the choice of the classifier. Interestingly enough, even if only these circled points were the part of the dataset, then the same classifier will be obtained. Although on one hand, one can argue that this is a good property of SVM, it turns that it can be very susceptible to the noise in the training data. Fig. 6.3(b) shows how by change of a single data point a very different classifier may be learned.

Thus, the notion of margin used in SVM makes it extremely sensitive to the noise in the data and is problematic when the data is non-separable. Researchers have got around this problem by introducing slack variables and ig-



(a)

(b)

Figure 6.2. (a) Possible linear classifiers that can classify the data without making error. (b) A possible classifier that will be chosen by Perceptron learning algorithm.

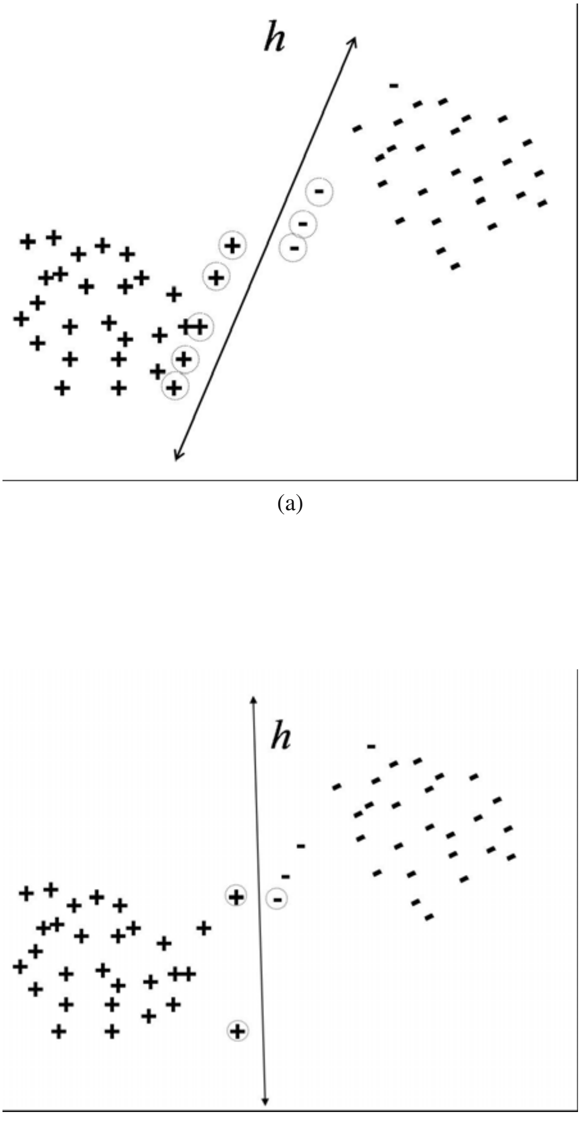


Figure 6.3. (a) The hyperplane that will be learned by SVM. The circled points correspond to the support vectors. (b) The learned hyperplane when a single datapoint is changed.

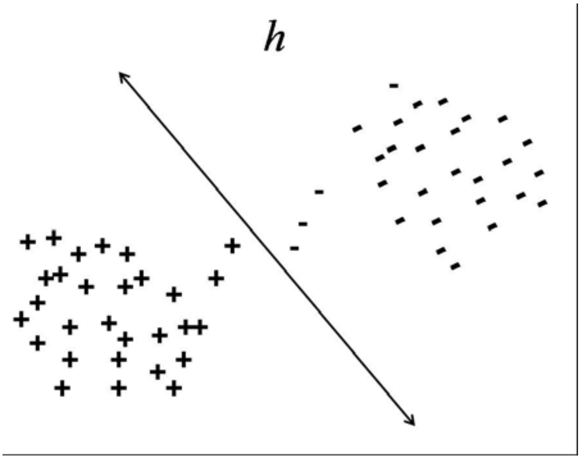


Figure 6.4. The hyperplane that may be a better choice then the one learned by SVM as in this case most of the data points (except for a few) are further apart from the hyperplane.

noring some of the closest point in an ad hoc fashion, but this extreme notion of margin still drives the optimization. Fig. 6.4 shows a hyperplane that tries to maximize the distance between all the datapoints and not just the closest points. This is the hyperplane that will be learned by the algorithm presented in the next section and our results on many standard data sets and some real world problems show this is indeed a better hyperplane.

4. The Margin Distribution Optimization (MDO) Algorithm

Along with generalization bounds presented in Chapter 3, our analysis provides guidance as to how to set up an optimization criterion for a linear classifier in order to optimize its generalization properties. We use this below to describe the Margin Distribution Optimization (MDO) algorithm.

The projection profile introduced can be viewed as assigning a weight to each point as a function of its distance from the hyperplane. The weight is the probability of making an error on that point when it is projected, along with the hyperplane, to some low dimensional space. Eqn. 3.4 relates this quantity, μ_k, directly to the generalization error. The smaller this quantity is, the smaller is the generalization error.

The new learning algorithm we propose, the Margin Distribution Optimization (MDO) algorithm, attempts to choose a classifier with the best projection profile (least projection error), while at the same time minimizing the classification error. This is achieved by formulating an optimization problem with two terms, one which depends on the projection profile and the second on the classification performance. For the linearly separable case, one can formulate the Lagrange equations as follows:

$$\mathcal{L}(h,\alpha) = \sum_{i=1}^{m} \exp\left(-\frac{k(h^T x_i)^2}{72||h||^2}\right) - \sum_{i=1}^{m} \alpha_i y_i h^T x_i + \sum_{i=1}^{m} \alpha_i. \tag{6.3}$$

Note that instead of employing the true projection profile, we use only the exponential term (which anyhow dominates for large enough values of k). The reason is that the second term is ill-behaved for points with extremely small margin.

Extending this to the non linearly separable case is straight forward, by introducing the slack variables. However, in our initial analysis, we adopt the following formulation:

$$\mathcal{L}(h,\alpha) = \sum_{i=1}^{m} \exp\left(-\frac{k(h^T x_i)^2}{72||h||^2}\right) - \sum_{i=1}^{m} \alpha_i \left(y_i h^T x_i + \xi_i ||h||\right) + \sum_{i=1}^{m} \alpha_i \tag{6.4}$$

where ξ_i is a positive number. Note that we are multiplying ξ_i with $||h||$.

This is done as now one can think of ξ_i as the slack in terms of the normalized margin (the data is already assumed to belong to unit ball in n dimensional space). One can either obtain the ξ_i by incorporating them in the optimization problem or use a search strategy over a holdout set. In our case, we assume that $\xi_i = c$ for all i and search for the best value of c and k (projection dimension) over a holdout set. Intuitively, it means that we allow for errors on points which are at a distance less than ξ_i from the hyperplane.

4.1 Comparison with SVM and Boosting

The difference between the above formulation and the SVM formulation [Burges, 1998] is primarily in the objective function (the first term in the above Lagrangian). In the SVM formulation the objective function minimizes the norm $||h||$ whereas here we minimize the projection profile.

It is interesting to note that boosting can also be thought of as a linear classifier (over the weak learners) which optimizes a somewhat similar cost function – $\sum_i \exp(-y_i h^T x_i)$ [Schapire and Singer, 1999] and therefore, we include it in our comparison below.

4.2 Computational Issues

The above optimization problem can be solved using the gradient descent algorithm. The gradients can be computed easily by taking the derivative with respect to each of the variables. However, the non-convexity of the objective function makes it a hard problem and the gradient descent in this case will only

Table 6.1. Results comparing the performance of Perceptron, SVM, boosting, and MDO. The first dataset is the synthetic data. The second dataset is the standard Compaq face dataset. Next four datasets are from UCI ML repository. The last two datasets are from the context sensitive spelling correction problem. As seen, in all cases, MDO's results are comparable to state of art results for these datasets.

Datasets	Perceptron	Boosting	SVM	MDO
Synthetic Data	73.25%	73.75%	78.50%	79.75%
Face Detection	89.5%	91.5%	91.5%	95.0%
Mushroom (UCI)	83.61%	83.77%	87.96%	89.14%
Liver (UCI)	56.88%	59.69%	60.94%	63.44%
Ionosphere (UCI)	81.90%	84.39%	84.84%	85.29%
Heart (UCI)	82.37%	83.05%	83.04 %	84.41%
peace & piece (Spelling)	70.50%	71.98%	73.95%	74.33%
being & begin (Spelling)	90.36%	91.33%	91.57%	92.05%

converge to a local minima. This implies that the solution obtained will not be the optimal solution and will depend highly on the starting point. There are two approaches that one can adopt:

1 try different starting points and choose the solution that is the best among the ones obtained;

2 Start from a good starting point.

We have adopted the latter approach in solving this optimization problem. Our approach first searches for a feasible solution (in the non-separable case – an admissible solution); maximizing the objective function using gradient descent in the space of feasible solution starts there. MDO starts by searching for the best hyperplane using the Perceptron algorithm (which is very fast). The learned classifier (output of the Perceptron algorithm) is then used as initial guess. At each step of the gradient descent, we make a move in the steepest direction, while ensuring the feasibility of the new solution (the constraints in terms of slack variables are satisfied). Typically, the algorithm converges in a small number of steps.

5. Experimental Evaluation

We did extensive evaluation of the algorithm on various datasets and compared its performance to SVM, boosting, and Perceptron learning algorithms.

We evaluated MDO on synthetic data (for the linearly separable case), two large scale real world problems – face detection [Pavlovic and Garg, 2001] and context sensitive spelling correction [Golding and Roth, 1999], and a number of datasets from UCI-ML repository. The classification performance is compared to SVM and boosting (over the same features). We use the same gradient descent algorithm and the same search strategies for ξ_i for both boosting and SVM. Essentially, by using the different objective function, we obtain different algorithms. For synthetic data, we randomly generated 1000 dimensional examples labeled according to a linear classifier (linearly separable data). The next experiment was done using face detection data [Pavlovic and Garg, 2001]. This is 756 dimensional data with 2000 examples. The spelling data is taken from the *pruned* data set of [Golding and Roth, 1999]. Two particular examples of context sensitive spelling correction *being & begin* and *peace & piece* are considered. For details on other dataset see [Murphy, 1994]. In all experiments, data was divided into three categories training (80%), holdout set (10%), and test set (10%). Five fold cross-validation was done and the average results are reported in Table 6.1. It is evident that MDO has consistently better performance than Perceptron, boosting, and SVM.

6. Conclusions

We have presented a new analysis method for linear learning algorithms that uses random projections and margin distribution analysis. The complexity measure (projection profile) presented in Chapter 3, has been used to develop a new learning algorithm for linear functions, which optimizes a measure directly related to the generalization error. The results are based on a novel use of the margin distribution of the data relative to the learned classifier, different than the typical use of the notion of margin in machine learning. Consequently, the resulting algorithm is not sensitive to small number of samples in determining the optimal hyperplane. Algorithmically, although we have given an implementation of the new algorithm MDO, one of the main direction of future research is to study it further, as well as to investigate other algorithmic implications of the ideas presented here.

Chapter 7

ALGORITHM: LEARNING THE STRUCTURE OF BAYESIAN NETWORK CLASSIFIERS

In Chapter 4, we presented a new analysis that showed under what conditions unlabeled data can be used in learning to improve classification performance. We showed that if these conditions are violated the use of unlabeled data can detrimental to the classification accuracy.

The goal of this chapter is to discuss the implications of this analysis to a specific type of classifiers, Bayesian networks. We investigate possible strategies for choosing a good graphical structure of the Bayesian networks and argue that in many problems it is necessary to search for such a structure. As a consequence, we propose a new structure learning algorithm that can utilize the unlabeled data to improve classification.

1. Introduction

Bayesian networks can represent joint distributions in an intuitive and efficient way; as such, Bayesian networks are naturally suited for classification. We can use a Bayesian network to compute the a-posteriori probability of a set of *labels* given the observable *features*, and then we classify the features with the most probable label.

A Bayesian network classifier represents dependencies among features and labels by a directed acyclic graph. This graph is the *structure* of the Bayesian network. Typically, Bayesian network classifiers are learned with a fixed structure — the paradigmatic example is the Naive Bayes classifier. More flexible learning methods allow Bayesian network classifiers to be selected from a small subset of possible structures — for example, the Tree-Augmented-Naive-Bayes (TAN) structures [Friedman et al., 1997]. After a structure is selected, the parameters of the classifier are usually learned using maximum likelihood estimation. In many cases, the selected structure does not match the structure that is generating the data, resulting in classifiers that cannot achieve the optimal Bayes error rate, even with infinite labeled training data (e.g., the Naive Bayes classifier).

The analysis in Chapter 4 indicates that the assumed structure is a key issue when learning Bayesian network classifiers with labeled and unlabeled data; an incorrect structure can have dire consequences. What we want to stress is this: However satisfactory a Naive Bayes (or similar) classifier may be in supervised learning, it is almost certainly a very sub-optimal solution in semi-supervised learning. The goal of this chapter is to discuss methods that can make *positive* use of unlabeled data when learning Bayesian Network classifiers.

Section 7.3 discusses the use of unlabeled data to switch between Bayesian networks that are simple and can be learned efficiently, namely Naive Bayes and TAN classifiers. In cases where relatively mild changes in structure still suffer from performance degradation from unlabeled data, there are different approaches that can be taken; discard the unlabeled data, try to learn the structure, or use the alternative of actively labeling some of the unlabeled data (Section 7.8). We discuss and compare different structure learning methods: likelihood score-based methods, and discusss independence-based methods (developed for semi-supervised learning by Marcelo Cirelo).

We also propose a stochastic structure search method based on Metropolis-Hastings sampling [Metropolis et al., 1953] over a measure induced by classification performance (Section 7.5). We argue that under certain conditions, the algorithm finds a Bayesian network that minimizes the classification error. We further use the VC dimension of Bayesian network classifiers to control the capacity and avoid overfitting to the training data during the search. Bayesian networks have an enormous advantage over less “structured” approaches; the existence of an underlying graphical structure allows us to explore the space of joint distributions in an organized fashion. Our search method seems to be the first that focuses solely on classification performance.

The experiments in Section 7.6 both for fully labeled sets and labeled and unlabeled sets, compare the different algorithms, pointing to their strengths and weaknesses.

2. Bayesian Network Classifiers

We consider classifiers that represent $p(C, \mathbf{X})$ using Bayesian networks [Pearl, 1988]. A Bayesian network is composed of a directed acyclic graph in which every node is associated with a variable X_i and with a conditional distribution $p(X_i|\Pi_i)$, where Π_i denotes the parents of X_i in the graph. The joint probability distribution is factored to the collection of conditional probability distribution of each node in the graph as:

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i|\Pi_i).$$

The directed acyclic graph is the *structure*, and the distributions $p(X_i|\Pi_i)$ represent the *parameters* of the network. Consider now that data generated by a distribution $p(C, \mathbf{X})$ are collected. We say that the assumed structure for a network, S', is *correct* when it is possible to find a distribution, $p(C, \mathbf{X}|S')$, that matches the distribution that generates data $p(C, \mathbf{X})$; otherwise, the structure is *incorrect*[1],[2].

A Bayesian network classifier is a *generative* classifier when the class variable is an ancestor of some or all features. A Bayesian network classifier is *diagnostic*, when the class variable has non of the features as descendants. As we are interested in using unlabeled data in learning the Bayesian network classifier, we restrict ourselves to generative classifiers, and exclude structures that are diagnostic.

Typically, Bayesian network classifiers are learned with a fixed structure – the paradigmatic example is the Naive Bayes classifier. More flexible learning methods allow Bayesian network classifiers to be selected from a small subset of possible structures – for example, the Tree-Augmented-Naive-Bayes structures [Friedman et al., 1997]. After a structure is selected, maximum likelihood is the common estimator for learning a network's parameters. Given missing data in our training set and for a fixed structure, we use the EM algorithm [Dempster et al., 1977] to learn the parameters of the network.

Given a Bayesian network classifier with parameter set Θ, the optimal classification rule under the maximum likelihood (ML) framework to classify an observed feature vector of n dimensions, $\mathbf{X} \in R^n$, to one of $|C|$ class labels, $c \in \{1, ..., |C|\}$, is given as:

$$\hat{c} = \underset{c}{argmax} \ P(\mathbf{X}|c; \Theta). \tag{7.1}$$

There are two design decisions when building Bayesian network classifiers. The first is to choose the structure of the network, which will determine the dependencies among the variables in the graph. The second is to determine the distribution of the features. The features can be discrete, in which case the distributions are probability mass functions. The features can also be continuous, in which case one typically has to choose a distribution, with the most common being the Gaussian distribution. Both these design decisions determine the parameter set Θ which defines the distribution needed to compute the decision function in Eq. (7.1).

In the following, we present the basic issues concerning the simple Bayesian networks models: Naive Bayes and TAN.

[1] These definitions follow directly from the definitions of correct and incorrect models described in Chapter 4.

[2] There is not necessarily a uniques correct structure, e.g., if a structure is correct (as defined above), all structures that are from the same Markov equivalent class are also correct since casuality is not an issue.

2.1 Naive Bayes Classifiers

Naive Bayes classifier is a probabilistic classifier in which the features are assumed independent given the class. Naive Bayes classifiers have a very good record in many classification problems, although the independence assumption is usually violated in practice. The reason for the Naive Bayes success as a classifier is attributed to the small number of parameters needed to be estimated. Recently, Garg and Roth [Garg and Roth, 2001b] showed using information theoretic arguments additional reasons for the success of Naive Bayes classifiers. An example of a Naive Bayes classifier is given in Figure 7.1.

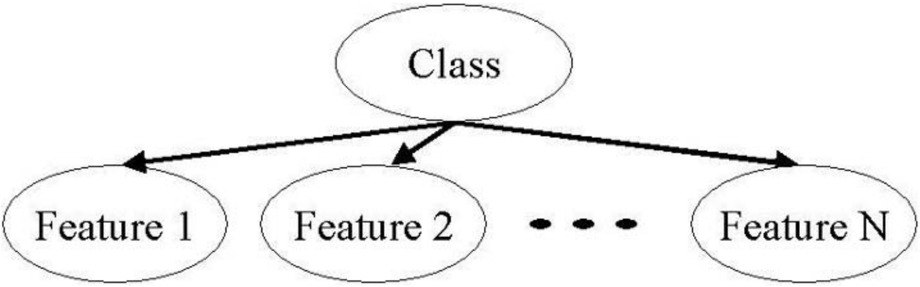


Figure 7.1. An example of a Naive Bayes classifier.

If the features in $\mathbf{X}$ are assumed to be independent of each other conditioned upon the class label c (the Naive Bayes framework), Eq. (7.1) reduces to:

$$\hat{c} = \underset{c}{argmax} \prod_{i=1}^{n} P(x_i|c;\Theta). \tag{7.2}$$

Now the problem is how to model $P(x_i|c;\Theta)$, which is the probability of feature x_i given the class label. In practice, the common assumption is that we have a Gaussian distribution and the ML can be used to obtain the estimate of the parameters (mean and variance). However, the Gaussian assumption is often invalid and the Cauchy distribution was proposed as an alternative model [Sebe and Lew, 2003]. This model was referred to as *Cauchy Naive Bayes*. The difficulty of this model is in estimating the parameters of the Cauchy distribution.

The Naive Bayes classifier was successful in many applications mainly due to its simplicity. Also, this type of classifier is working well even if there is not too much training data. However, the strong independence assumption may seem unreasonable in some cases. Therefore, when sufficient training data is available we want to learn and to use the dependencies present in the data.

2.2 Tree-Augmented Naive Bayes Classifiers

In the TAN classifier structure the class node has no parents and each feature has as parents the class node and at most one other feature, such that the result is a tree structure for the features (see Figure 7.2). Friedman et al. [Friedman et al., 1997] proposed using the TAN model as a classifier, to enhance the performance over the simple Naive Bayes classifier. TAN models are more complicated than the Naive Bayes, but are not fully connected graphs. The existence of an efficient algorithm to compute the best TAN model makes it a good candidate in the search for a better structure over the simple NB.

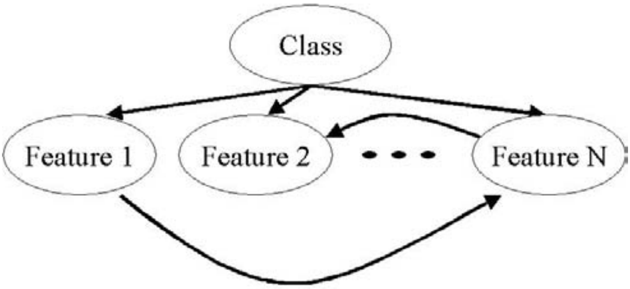


Figure 7.2. An example of a TAN classifier.

Learning the TAN classifier is more complicated. In this case, we do not fix the structure of the Bayesian network, but we try to find the TAN structure that maximizes the likelihood function given the training data out of all possible TAN structures.

In general, searching for the best structure has no efficient solution, however, searching for the best TAN structure does have one. The method is using the modified Chow-Liu algorithm [Chow and Liu, 1968] for constructing tree augmented Bayesian networks [Friedman et al., 1997]. The algorithm finds the tree structure among the features that maximizes the likelihood of the data by computation of the pairwise class conditional mutual information among the features and building a maximum weighted spanning tree using the pairwise mutual information as the weights of the arcs in the tree. The problem of finding a maximum weighted spanning is defined as finding the set of arcs connecting the features such that the resultant graph is a tree and the sum of the weights of the arcs is maximized. There have been several algorithms proposed for building a maximum weighted spanning tree [Cormen et al., 1990] and in our implementation we use the Kruskal algorithm described in Box 7.1.

The five steps of the TAN algorithm are described in Box 7.2. This procedure ensures to find the TAN model that maximizes the likelihood of the data we have. The algorithm is computed in polynomial time ($O(n^2 logN)$, with N being the number of instances and n the number of features).

Box 7.1 (Kruskal's Maximum Weighted Spanning Tree Algorithm)

Consider an undirected graph with n vertices and m edges, where each edge (u, v) connecting the vertices u and v, has an associated positive weight $w_{(u,v)}$. To construct the maximum weighted spanning tree graph follow the following steps:

1. Create an empty set of edges called *spanningTree*.
2. For each vertex v in the graph, create a set containing v.
3. Sort all edges in the graph using the weights in the edges from highest to lowest.
4. In order of the sorted edges, for each edge (u, v) if the set that contains u is different from the set that contains v:
   - Put the edge (u, v) in *spanningTree*
   - Make u and v belong to the same set (union of sets).
5. *spanningTree* contains all the edges in the maximum weighted spanning tree.

Box 7.2 (TAN learning algorithm)

1. Compute the class conditional pair-wise mutual information between each pair of features, (X_i, X_j) for all $i, j \in \{1, ..., n\}$,

$$I_P(X_i, X_j|C) = \sum_{X_i,X_j,C} P(x_i, x_j, c) \log \frac{P(x_i, x_j|c)}{P(x_i|c)P(x_j|c)}, i \neq j.$$

2. Build a complete undirected graph in which each vertex is a variable, and the weight of each edge is the mutual information computed in Step 1.
3. Build a maximum weighted spanning tree (MWST) (see Box 7.1).
4. Transform the undirected MWST of Step 3 to a directed graph by choosing a root node and pointing the arrows of all edges away from the root.
5. Make the class node the parent of all the feature nodes in the directed graph of Step 4.

The learning algorithm for the TAN classifier as proposed by Friedman et al. [Friedman et al., 1997] relies on computations of the class conditional mutual information of discrete features. In our problem, the features are continuous, and computation of the mutual information for a general distribution is very complicated. However, if we assume that the features are Gaussian, computation of the conditional mutual information is feasible and is given by (see Box 7.3 for details):

$$I(X_i, X_j|C) = -\frac{1}{2}\sum_{c=1}^{|C|} P(C=c)\log(1-\rho^2_{(ij)|c}), \tag{7.3}$$

where $\rho_{(ij)|c}$ is the correlation coefficient between X_i and X_j given the class label c. We replace the expression for the mutual information in Step 1 of the TAN algorithm with the expression in Equation (7.3), to find the maximum likelihood Gaussian-TAN classifier.

The full joint distribution of the Gaussian-TAN model can be written as:

$$p(c, x_1, x_2, ..., x_n) = p(c)\prod_{i=1}^{n} p(x_i|\Pi_i, c), \tag{7.4}$$

where Π_i is the feature that is the additional parent of feature x_i. Π_i is empty for the root feature in the directed tree graph of Step 4 in the Kruskal's algorithm.

Using the Gaussian assumption, the probability density functions (pdf's) of the distribution in the product above are:

$$p(X_i = x_i|\Pi_i, C=c) = N_c(\mu_{x_i} + a\cdot\Pi_i, \sigma^2_{x_i}\cdot(1-\rho^2)), \tag{7.5}$$

where $N_c(\mu, \sigma^2)$ refers to the Gaussian distribution with mean and variance given that the class is c, $\mu_{x_i}, \sigma^2_{x_i}$ are the mean and variance of the feature x_i,

$$\rho = \frac{COV(x_i, \Pi_i)}{\sigma_{x_i}\sigma_{\Pi_i}}$$

is the correlation coefficient between x_i and Π_i, and

$$a = \frac{COV(x_i, \Pi_i)}{\sigma^2_{\Pi_i}}.$$

For further details on the derivation of the parameters see Box 7.3.

Box 7.3 (Gaussian-TAN Parameters Computation)

The mutual information between continuous random variables, X, Y is given as [Kay, 1990; Starks and Woods, 1994]:

$$I(X,Y) = \int\int p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right)dxdy = H(x) + H(y) - H(x,y)$$

where $H(\cdot)$ is the differential entropy, analogous to the entropy of discrete variables, defined as:

$$H(Z) = -\int p(z)\log p(z)dz. \tag{7.6}$$

Here $p(z)$ is the probability density function of Z and the integral is over all dimensions in z.

For a Gaussian random vector Z of N dimensions with covariance matrix Σ, by inserting the Gaussian pdf to Eq. (7.6) and taking the integral, we get that the differential entropy of Z is:

$$H(Z) = \frac{1}{2}\log\left((2\pi e)^N|\Sigma|\right) \tag{7.7}$$

where $|\Sigma|$ is the determinant of Σ.

Suppose now that X and Y are jointly Gaussian. Then,

$$p(X,Y) \sim N\left(\begin{bmatrix}\mu_X \\ \mu_Y\end{bmatrix}, \Sigma_{XY}\right) \tag{7.8}$$

where Σ_{XY} is the covariance matrix given as:

$$\Sigma_{XY} = \begin{bmatrix}\sigma_X^2 & COV(X,Y) \\ COV(X,Y) & \sigma_Y^2\end{bmatrix}. \tag{7.9}$$

Using Eqs. (7.7) and (7.6) we get that the mutual information of X and Y is given by:

$$\begin{aligned} I(X,Y) &= -\frac{1}{2}\log\left(\frac{\sigma_X^2\sigma_Y^2}{\sigma_X^2\sigma_Y^2 - COV(X,Y)^2}\right) \\ &= -\frac{1}{2}\log\left(\frac{1}{1-\frac{COV(X,Y)^2}{\sigma_X^2\sigma_Y^2}}\right) = -\frac{1}{2}\log\left(\frac{1}{1-\rho_{XY}^2}\right) \end{aligned} \tag{7.10}$$

where $\rho_{XY} = \frac{COV(X,Y)^2}{\sigma_X^2\sigma_Y^2}$ is the correlation coefficient between X and Y.

In the TAN classifiers, the class is the parent of all features, and the features are Gaussian given a class label. Thus all the results above apply with an understanding that the distributions are conditioned on the class label (which is omitted for clarity). The class conditional mutual information between the pair X and Y is derived as follows:

$$\begin{aligned} I(X,Y|C) &= \sum_{c=1}^{|C|} \int\int p(x,y,c) \log \left(\frac{p(x,y|c)}{p(x|c)p(y|c)} \right) dxdy \\ &= \sum_{c=1}^{|C|} \int\int p(c)p(x,y|c) \log \left(\frac{p(x,y|c)}{p(x|c)p(y|c)} \right) \\ &= \sum_{c=1}^{|C|} p(c) I(X,Y|C=c) \\ &= -\frac{1}{2} \sum_{c=1}^{|C|} p(c) \log \left(\frac{1}{1-\rho_{XY|c}^2} \right). \end{aligned} \tag{7.11}$$

After finding the TAN structure, suppose that we find that feature X is the parent of Y. Given the class label, X and Y are jointly Gaussian with mean vector and covariance as defined in Eqs. (7.8) and (7.9) (again omitting the conditioning on the class variable for clarity). Since X is the parent of Y, we are interested in finding the parameters of the conditional distribution $p(Y|X)$ as a function of the parameters of the joint distribution. Because X and Y are jointly Gaussian, $Y|X$ is also Gaussian. Using $p(X,Y) = p(X)p(Y|X)$ and the Gaussian pdf, after some manipulations we get:

$$\begin{aligned} p(Y|X) &= \frac{p(X,Y)}{p(X)} \\ &= \frac{1}{(2\pi\sigma_Y^2(1-\rho_{XY}^2))^{1/2}} \exp \left(-\frac{(y-\mu_Y-ax)^2}{2\sigma_Y^2(1-\rho_{XY}^2)} \right) \\ &= N\left(\mu_Y + ax, \sigma_Y^2(1-\rho_{XY}^2)\right) \end{aligned} \tag{7.12}$$

where $a = \frac{COV(X,Y)}{\sigma_X^2}$.

After learning the structure, the Gaussian-TAN classifier's added complexity compared to the Naive Bayes classifier is small; there are $|C| \cdot (n-1)$ extra parameters to estimate (the covariances between features and their parents). For learning the structure, all pairwise mutual information are estimated using the estimates for the covariances.

3. Switching between Models: Naive Bayes and TAN Classifiers

The conclusion of Chapter 4 indicates the importance of obtaining the correct structure when using unlabeled data in learning the classifier. If the correct structure is obtained, unlabeled data improve a classifier; otherwise, unlabeled data can actually degrade performance. Somewhat surprisingly, the option of searching for structures has not been proposed by researchers that have previously witnessed the performance degradation. Apparently, performance degradation was attributed to unpredictable, stochastic disturbances in modeling assumptions, and not to mistakes in the underlying structure — something that can be detected and fixed.

One attempt to overcome the performance degradation from unlabeled data could be to switch models as soon as degradation is detected. Suppose then that we learn a classifier with labeled data only, and we observe a degradation in performance when the classifier is learned with labeled and unlabeled data. We can switch to a more complex structure at that point. As we saw in Chapter 4, bias and variance play an important role in the utilization of unlabeled data. To preserve the balance between the bias from the true distribution and the variance we might want to use a small subset of simple models which can be learned efficiently.

We start with the simplest generative structure, the Naive Bayes. Despite having a non-zero classification bias, the Naive-Bayes classifier performs well for many cases, *when trained with labeled data*. The success is explained in the literature using several arguments; e.g., trade-offs between classification bias and variance when learning with scarce data [Friedman, 1997] and tendency of many distributions to be close (in the Kullback-Leibler sense) to the product distribution of the Naive Bayes classifier [Garg and Roth, 2001b]. However, in semi-supervised learning, the same success is not always observed (see experiments in this chapter and in Chapter 4).

If a problem is such that Naive Bayes classifiers suffer from performance degradation with unlabeled data, we should then switch to a larger family of models. We can decide to switch using the test described in Section 4.6.4. The most promising such family is represented by TAN classifiers, in which the class variable is a parent of all of the observables, and the observables are connected so as to form a tree. Friedman et al. [Friedman et al., 1997] showed that learning the most likely TAN structure can be done efficiently using the Chow-Liu algorithm [Chow and Liu, 1968].

The TAN algorithm assumes that there is no missing data. To solve the maximization problem when there are missing labels, it is possible to develop an EM algorithm that finds the best TAN given both labeled and unlabeled data. Meila [Meila, 1999] developed the EM equations for the task of building the best set of minimum spanning tree for a classification problem. In her setup, there are no labeled records and the number of classes can vary. The problem she solves is basically a clustering problem and is not directly related to the semi-supervised learning problem, but the EM algorithm she developed applies directly to the semi-supervised learning problem. The EM algorithm for TAN models given labeled and unlabeled data follows directly from Meila's [Meila, 1999] *MixTreeS* algorithm; we call the resulting scheme EM-TAN [Cohen et al., 2002a]. The general steps of EM-TAN are shown in Box 7.4. The algorithm enjoys the efficiency of the supervised TAN algorithm, while guaranteeing convergence to a local maximum of the likelihood function.

Box 7.4 (EM-TAN Algorithm)

- An initial TAN structure and parameters are set. The class variable is always the root node. The initial TAN is either arbitrarily chosen or is found by estimating the TAN model with only the labeled records and the TAN algorithm (Box 7.2).
- Iterate until the change in the likelihood between the current iteration and previous falls under a threshold:
  - E-Step: Compute expected value of the missing labels using the current model parameters.
  - M-Step:
    - Compute the class conditional mutual information between all pairs of feature variables using the fractional counts from the
    - Construct the MWST using the Chow-Liu algorithm.
    - Compute all of the parameters of the new model given the MWST found in the previous step.

We have observed that EM-TAN produces classifiers that in practice regularly surpass Naive Bayes classifiers. Still, performance degradation can still occur both for Naive Bayes and TAN (as actually observed in Section 7.6). In such cases, we are faced with several options. The first is to discard the unlabeled data and use only the available labeled samples. The other options are discussed in the next sections.

4. Learning the Structure of Bayesian Network Classifiers: Existing Approaches

If we observe performance degradation, we may try to find the "correct" structure for our Bayesian network classifier – if we do so, we can profitably use unlabeled data. Alas, learning Bayesian network structure is not a trivial task. In this section, we investigate the behavior of structure learning algorithms in the context of semi-supervised learning, presenting new algorithms where needed, and deriving new techniques that improve on existing methods. Experiments validating such claims are presented in Section 7.6.

4.1 Independence-based Methods

The first class of structure learning methods we consider is the class of independence-based methods, also known as constraint-based or test-based methods. There are several such algorithms; a relevant subset is composed of the PC algorithm [Spirtes et al., 2000], the IC algorithm [Pearl, 2000], and the Cheng-Bell-Liu algorithms (CBL1 and CBL2) [Cheng et al., 1997][3]. All of them can obtain the correct structure if there are fully reliable independence tests available; however not all of them are appropriate for classification. For example, the PC algorithm starts with a fully connected network, and has the tendency to generate structures that are "too dense" (consequently requiring many parameters to be learned, negatively affecting the variance of estimated quantities and increasing the classification error).

The CBL1 and CBL2 algorithms seem particularly well-suited for classification, as they strive to keep the number of edges in the Bayesian networks as small as possible. They won the ACM KDD cup 2001 data mining Competition, a considerable feat. Moreover, the performance of CBL1 on labeled data only has been reported to surpass the performance of TAN, even with arbitrary node orderings [Cheng and Greiner, 1999]. Conceptually CBL1 and CBL2 are similar, with CBL1 requiring an ordering to start. We used conditional independence (CI) tests based on mutual information: we declare variables X and Y to be independent given variable Z when their mutual information conditional on Z is smaller than a constant ϵ, which we set to 0.01.

A few modifications are necessary to adapt CBL1 and CBL2 for semi-supervised learning. First, the algorithms are started with a Naive Bayes classifier, and in CBL1 arcs from the class variable to observed variables are allowed

[3] The work on the CBL algorithms was done by primarily by Marcelo C. Cirelo who investigated and implemented the algorithm. This subsection is a summary of his work and conclusions.

to be removed, leading to some restricted forms of feature selection. More importantly, a simple method to generate orderings for CBL1 is developed by generating a fixed number of random orderings, and running the algorithm for all of them. Because CBL1 is quite fast, hundreds of candidate orderings are easily tested, selecting the one that produces the best classifier (using either testing data or cross-validation to select the classifier, depending on the amount of available labeled data).

Because independence-based algorithms like CBL1 do not explicitly optimize a metric, they cannot handle unlabeled data directly through an optimization scheme like EM. To handle unlabeled data, we chose the strategy presented in Box 7.5 (denoted as EM-CBL). It should be noted that such a scheme, however intuitively reasonable, has no convergence guarantees; one test even displayed oscillating behavior.

Box 7.5 (EM-CBL Algorithm)

- Start by learning a Bayesian network with the available labeled data.
- Repeat until two subsequent networks are identical:
  - Use EM to process unlabeled data.
  - Use independence tests with the "probabilistic labels" generated by EM, to obtain a new structure.

Despite such difficulties, EM-CBL1 has been observed to actually improve the performance obtained with EM-TAN in many problems (see Section 7.6). This apparent victory must be taken carefully though: the algorithm takes much more computational effort than EM-TAN, and its improvement over EM-TAN is only marginal. Moreover, the algorithm relies on the computation of mutual information with the "probabilistic labels" generated by EM; such a method has been observed to lead to unreliable CI tests. Given the fact that independence-based algorithms all depend critically on these tests, the lack of robustness of such tests creates difficulties for EM-CBL1 in several classification problems.

The EM-CBL2 was also tested (produced by running the algorithm described in the previous paragraph, with CBL1 replaced by CBL2). The algorithm is fast, but it is extremely sensitive to independence tests; in many situations it cannot find sensible orientation to edges and in some cases it makes conflicting decisions. EM-CBL2 has been observed to be consistently *worse* than EM-TAN, hence it was not explored further.

To conclude, experience shows that the use of independence-based methods in semi-supervised learning is not promising.

4.2 Likelihood and Bayesian Score-based Methods

Here we turn to a different family of algorithms, those based on scores. At the heart of most score based methods is the likelihood of the training data. To avoid overfitting the model to the data, the likelihood is offset by a complexity penalty term, such as the minimum description length (MDL), Bayesian information criterion (BIC), and others. A good comparison of the different methods is found in [van Allen and Greiner, 2000]. Most existing methods cannot, in their present form, handle missing data in general and unlabeled data in particular. The structural EM (SEM) algorithm [Friedman, 1998] is one attempt to learn structure with missing data. The algorithm attempts to maximize the Bayesian score using an EM-like scheme in the space of structures and parameters; the method performs an always-increasing search in the space of structures, but does not guarantee the attainment of even a local maximum. The algorithms using other scores could most likely be extended to handle unlabeled data in much the same way as the SEM algorithm.

When learning the structure of a classifier, score based structure learning approaches (such as BIC and MDL) have been strongly criticized. The problem is that with finite amounts of data, the a-posteriori probability of the class variable can have a small effect on the score, that is dominated by the marginal of the observables, therefore leading to poor classifiers [Friedman et al., 1997; Greiner and Zhou, 2002]. Friedman et al. [Friedman et al., 1997] showed that TAN surpasses score based methods for the fully labeled case, *when learning classifiers*. The point is that with unlabeled data, score based methods such as SEM are likely to go astray even more than what has been reported in the supervised case; the marginal of the observables further dominates the likelihood portion of the score as the ratio of unlabeled data increases.

Bayesian approaches to structure learning have also been proposed in [Friedman and Koller, 2000; Madigan and York, 1995]. Madigan and York [Madigan and York, 1995] construct a Markov Chain Monte Carlo (MCMC) over the space of possible structures, with the stationary distribution being the posterior of the structures given the data. Metropolis sampling-Hastings [Metropolis et al., 1953] is used to sample from the posterior distribution. Friedman and Koller [Friedman and Koller, 2000] use a two step method in their sampling – first they sample from the distribution over the ordering of the variables followed by exact computation of the desired posterior given the ordering. As with likelihood scores, we can expect these two methods to face difficulties when learning classifiers, since they focus on the joint distribution given the data, and not on the classification error or the a-posteriori probability of the class variable.

5. Classification Driven Stochastic Structure Search

Both the score-based and independence-based methods try to find the correct structure of the Bayesian network, but fail to do so because there is not enough data for either reliable independence tests or for a search that yields a good classifier. Consider the following alternative. As we are interested in finding a structure that performs well as a classifier, it would be natural to design algorithms that use classification error as the guide for structure learning. Here, we can further leverage on the properties of semi-supervised learning: we know that unlabeled data can indicate incorrect structure through degradation of classification performance, and we also know that classification performance improves with the correct structure. Thus, a structure with higher classification accuracy over another indicates an improvement towards finding the optimal classifier.

5.1 Stochastic Structure Search Algorithm

To learn structure using classification error, we must adopt a strategy of searching through the space of all structures in an efficient manner while avoiding local maxima. In this section, we propose a method that can effectively search for better structures *with an explicit focus on classification.* We essentially need to find a search strategy that can efficiently search through the space of structures. As we have no simple closed-form expression that relates structure with classification error, it would be difficult to design a gradient descent algorithm or a similar iterative method. Even if we did that, a gradient search algorithm would be likely to find a local minimum because of the size of the search space.

First we define a measure over the space of structures which we want to maximize:

DEFINITION 7.1 *The* inverse error measure *for structure S' is*

$$inv_e(S') = \frac{\frac{1}{p_{S'}(\hat{c}(X)\neq C)}}{\sum_S \frac{1}{p_S(\hat{c}(X)\neq C)}}, \tag{7.13}$$

where the summation is over the space of possible structures and $p_S(\hat{c}(X) \neq C)$ is the probability of error of the best classifier learned with structure S.

We use Metropolis-Hastings sampling [Metropolis et al., 1953] to generate samples from the inverse error measure, without having to ever compute it for all possible structures. For constructing the Metropolis-Hastings sampling, we define a neighborhood of a structure as the set of directed acyclic graphs to which we can transit in the next step. Transition is done using a predefined set of possible changes to the structure; at each transition a change consists of a single edge addition, removal or reversal. We define the acceptance probability of a candidate structure, S_{new}, to replace a previous structure, S_t as follows:

$$\min\left(1, \left(\frac{inv_e(S^{new})}{inv_e(S^t)}\right)^{1/T} \frac{q(S^t|S^{new})}{q(S^{new}|S^t)}\right) = \min\left(1, \left(\frac{p^t_{error}}{p^{new}_{error}}\right)^{1/T} \frac{N_t}{N_{new}}\right), \quad (7.14)$$

where $q(S'|S)$ is the transition probability from S to S' and N_t and N_{new} are the sizes of the neighborhoods of S_t and S_{new} respectively; this choice corresponds to equal probability of transition to each member in the neighborhood of a structure. This choice of neighborhood and transition probability creates a Markov chain which is aperiodic and irreducible, thus satisfying the Markov chain Monte Carlo (MCMC) conditions [Madigan and York, 1995]. The algorithm, which we name stochastic structure search (SSS), is presented in Box 7.6.

Box 7.6 (Stochastic Structure Search Algorithm)

- Fix the network structure to some initial structure, S_0.
- Estimate the parameters of the structure S_0 and compute the probability of error p^0_{error}.
- Set $t = 0$.
- Repeat, until a maximum number of iterations is reached ($MaxIter$):
  - Sample a new structure S_{new}, from the neighborhood of S_t uniformly, with probability $1/N_t$.
  - Learn the parameters of the new structure using maximum likelihood estimation. Compute the probability of error of the new classifier, p^{new}_{error}.
  - Accept S_{new} with probability given in Eq.(7.14).
  - If S_{new} is accepted, set $S_{t+1} = S_{new}$ and $p^{t+1}_{error} = p^{new}_{error}$ and change T according to the temperature decrease schedule. Otherwise $S_{t+1} = S_t$.
  - $t = t + 1$.
- return the structure S_j, such that $j = \underset{0 \leq j \leq MaxIter}{argmin} (p^j_{error})$.

We add T as a temperature factor in the acceptance probability. Roughly speaking, T close to 1 would allow acceptance of more structures with higher probability of error than previous structures. T close to 0 mostly allows acceptance of structures that improve probability of error. A fixed T amounts to changing the distribution being sampled by the MCMC, while a decreasing T is a simulated annealing run, aimed at finding the maximum of the inverse error measures. The rate of decrease of the temperature determines the rate of convergence. Asymptotically in the number of data, a logarithmic decrease of T guarantees convergence to a global maximum with probability that tends to one [Hajek, 1988].

The SSS algorithm, with a logarithmic cooling schedule T, can find a structure that is close to minimum probability of error. There are two caveats though. First, the logarithmic cooling schedule is very slow. Second, we never have access to the true probability of error for each structure – we estimate it from a limited pool of training data. To avoid the problem of overfitting we can take several approaches. Cross-validation can be performed by splitting the labeled training set to smaller sets. However, this approach can significantly slow down the search, and is suitable only if the labeled training set is moderately large. A different approach is to change the the error measure using known bounds on the empirical classification error, which account for model complexity. We describe such an approach in the next section.

5.2 Adding VC Bound Factor to the Empirical Error Measure

In practice, we do not have access to the exact probability of error, p^S_{error}, for any given structure. Instead, we use the empirical error over the training data:

$$\hat{p}^S_{error} = \frac{1}{n}\sum_{i=1}^{n} I(\hat{c}_i, c_i), \tag{7.15}$$

where $I(\hat{c}, c)$ is 1 for $\hat{c} = c$ and 0, otherwise. $\hat{c}_i$ is the prediction of the class given the learned classifier, c_i is the true label, and n is the number of training data. Using this approximation to the error, any search method might eventually overfit the training data, finding the global optimum of the empirical error, $\hat{p}_{error}$, but resulting in a poor classifier for unseen data.

We use the multiplicative penalty term derived from structural risk minimization to define a modified error term for use in Eqs. (7.13) and (7.14):

$$(\hat{p}^S_{error})^{mod} = \frac{\hat{p}^S_{error}}{1 - c \cdot \sqrt{\frac{h_S(log\frac{2n}{h_S}+1)-log(\eta/4)}{n}}}, \tag{7.16}$$

Table 7.1. Classification results (in %) for labeled only tests, with 95% confidence intervals.

Data Set	# Training	# Test	SSS	NB	TAN
Shuttle	43500	14500	99.93±0.03	99.79±0.04	99.86±0.03
Satimage	4435	2000	87.70±0.73	81.80±0.86	86.15±0.77
Adult	30162	15060	85.94±0.28	83.85±0.29	85.08±0.29
Chess	2130	1066	96.53±0.59	88.38±0.99	91.93±0.85

where h_S is the Vapnik-Chervonenkis (VC) dimension of the classifier with structure S, n is the number of training records, η and c are between 0 and 1.

To approximate the VC dimension, we use: $h_S \propto N_S$, where N_S is the number of (free) parameters in the Markov blanket of the class variable in the network, assuming that all variables are discrete. We point the reader to [Roth, 1999], in which it was shown that the VC dimension of a Naive Bayes classifier is linearly proportional to the number of parameters. It is possible to extend this result to networks where the features are all descendants of the class variable. For more general networks, features that are not in the Markov blanket of the class variable cannot affect its value in classification (assuming there are no missing values for any feature), justifying the above approximation. In our initial experiments, we found that the multiplicative penalty outperformed the holdout method and the MDL and BIC complexity measures.

6. Experiments

The experiments test the SSS algorithm and compare it to the other approaches. First we investigate the performance using only labeled data with datasets from the UCI machine learning repository [Blake and Merz, 1998], validating that it is performing correct in general. We then removed the labels of most of the data randomly and learn with both labeled and unlabeled data. All of the results we present were obtained on independent test sets, which were *not* used in during the learning phase. For the SSS algorithm we used either the Naive-Bayes or TAN structure as the starting point network and use 300-1000 iterations of the SSS algorithm.

6.1 Results with Labeled Data

Table 7.1 shows the classification results with fully labeled data for a number of datasets. For each dataset, we compare the classifier obtained with our search method to Naive Bayes and TAN classifiers. Results of other classifiers on most of these datasets have been reported in [Friedman et al., 1997], and are comparable to the performance of the SSS algorithm. We see that for all the datasets, the structure search algorithm outperformed NB and TAN. The

Table 7.2. The datasets used in the experiments with labeled and unlabeled data.

Dataset	Train		Test
	# labeled	# unlabeled	
TAN artificial	300	30000	50000
Satimage	600	3835	2000
Shuttle	100	43400	14500
Adult	6000	24163	15060
Chess	150	1980	1060

most dramatic increase occurred for the Chess dataset, where the performance improved to over 96% accuracy.

6.2 Results with Labeled and Unlabeled Data

To evaluate structure learning methods with labeled and unlabeled data, we started with an empirical study involving simulated data. We artificially generated data to investigate:

1 whether the SSS algorithm finds a structure that is close to the structure that generated the data, and

2 whether the algorithm uses unlabeled data to improve the classification performance.

A typical result is as follows. We generated data from a TAN structure with 10 features. The dataset consisted of 300 labeled and 30000 unlabeled records (first row of Table 7.2). We first estimated the Bayes error rate by learning with the correct structure and with a very large fully labeled dataset. We obtained a classification accuracy of 92.49%. We learned one Naive Bayes classifier only with the labeled records, and another classifier with both labeled and unlabeled records; likewise, we learned a TAN classifier only with the labeled records, and another one with both labeled and unlabeled records, using the EM-TAN algorithm; and finally, we learned a Bayesian network classifier with our SSS algorithm using both labeled and unlabeled records. The results are presented in the first row of Table 7.3. With the correct structure, adding unlabeled data improves performance significantly (columns TAN-L and EM-TAN). Note that adding unlabeled data degraded the performance from 16% error to 40% error when we learned the Naive Bayes classifier. The structure search algorithm comes close to the performance of the classifier learned with the correct structure. Figure 7.3(a) shows the changes in the test and train error during the search process. The graph shows the first 600 moves of the search, initialized with the Naive Bayes structure. The error usually decreases as new

Table 7.3. Classification results (in %) for Naive Bayes classifier learned with labeled data only (NB-L), Naive Bayes classifier learned with labeled and unlabeled data (EM-NB), TAN classifier learned with labeled data only (TAN-L), TAN classifier learned with labeled and unlabeled data (EM-TAN), the EM-CBL1 algorithm, and stochastic structure search with labeled and unlabeled data (SSS).

Dataset	NB-L	EM-NB	TAN-L	EM-TAN	EM-CBL1	SSS
TAN artificial	83.4±0.2	59.2±0.2	90.9±0.1	91.9±0.1	N/A	91.1±0.1
Satimage	81.7±0.9	77.5±0.9	**83.5±0.8**	81.0±0.9	**83.5±0.8**	83.4±0.8
Shuttle	82.4±0.3	76.1±0.4	81.2±0.3	90.5±0.2	91.8±0.2	**96.3±0.2**
Adult	83.9±0.3	73.1±0.4	84.7±0.3	80.0±0.3	82.7±0.3	**85.0±0.3**
Chess	79.8±1.2	62.1±1.5	**87.0±1.0**	71.2±1.4	81.0±1.2	76.0±1.3

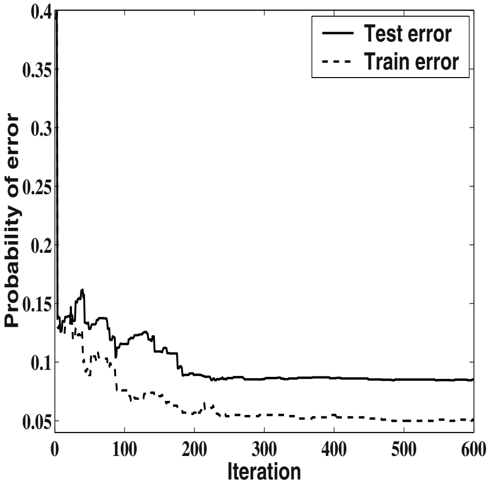
structures are accepted; occasionally we see an increase in the error allowed by Metropolis-Hastings sampling.

Next, we performed experiments with some of the UCI datasets, using relatively small labeled sets and large unlabeled sets (Table 7.2). The results in Table 7.3 suggest that structure learning holds the most promise in utilizing the unlabeled data. There is no clear 'winner' approach, although SSS yields better results in most cases. We see performance degradation with NB for every dataset. EM-TAN can sometimes improve performance over TAN with just labeled data (Shuttle). With the Chess dataset, discarding the unlabeled data and using only TAN seems the best approach.

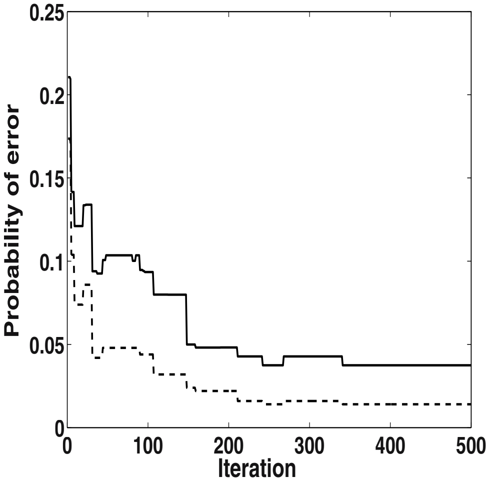
Illustrating the iterations of the SSS algorithm, Figure 7.3(b) shows the changes in error for the shuttle datasets. The Bayesian network structure learned with the SSS algorithm for the Shuttle database is shown in Figure 7.4.

As discussed in Section 7.4.2, likelihood based score methods are not promising when learning with labeled and unlabeled data. We illustrate the possible shortcomings of these methods with experiments on the databases discussed above. For all databases we learned the structure and parameters using the K2 [Cooper and Herskovits, 1992] and MCMC [Madigan and York, 1995] (driven by the Bayesian score) algorithms with the fully labeled datasets[4]. After this step, we learned the parameters of the same structure with the mixed labeled and unlabeled training sets as described in Table 7.3. To reduce the possibility of convergence of EM to a local maxima, the parameters learned with the fully labeled learning are used as the starting point for the EM algorithm. We intentionally assist the algorithms to make the point of their shortcomings. Table 7.4 shows the classification results for both cases (fully labeled

[4] Learning the structure was performed with the Bayesian network toolbox for Matlab, written by Dr. Kevin Murphy.



(a)



(b)

Figure 7.3. Train and test error during the structure search for the artificial data (a) and shuttle data (b) for the labeled and unlabeled data experiments.

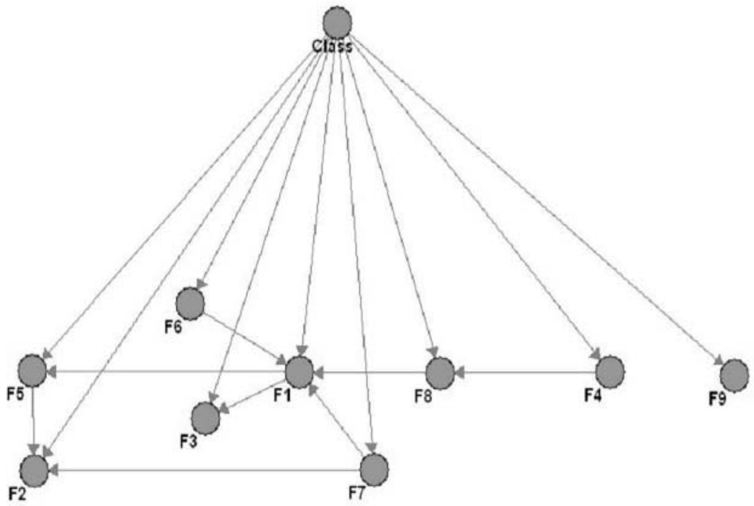


Figure 7.4. Bayesian network structure learned for the Shuttle database.

Table 7.4. Classification results (in %) for the K2 and MCMC structure learning. xx-L means using all the labeled data (Table 7.1). xx-LUL means using the labeled and unlabeled data (Table 7.2).

Data Set	K2-L	K2-LUL	MCMC-L	MCMC-LUL
Shuttle	96.4±0.2	86.8±0.3	N/A	N/A
Satimage	82.0±0.9	77.3±0.9	83.3±0.8	64.4±1.1
Adult	84.2±0.3	84.1±0.3	82.6±0.3	77.6±0.3
Chess	94.3±0.7	88.6±1.0	95.3±0.7	88.7±1.0

and partially labeled). The classification results with the fully labeled sets are comparable to the results in Table 7.1. With labeled and unlabeled data, we see that the for most cases, the full potential of unlabeled data was not utilized, even with the extensive ’help’ to the algorithms. In addition, with the exception of the Chess database, other results are inferior to those shown in Table 7.3.

7. Should Unlabeled Data Be Weighed Differently?

We see that in all the experiments, unlabeled data degraded the performance for the Naive Bayes structure. An interesting strategy, suggested by Nigam et al. [Nigam et al., 2000] was to change the weight of the unlabeled data (reducing their effect on the likelihood). The basic idea in Nigam et al’s estimators is to produce a modified log-likelihood that is of the form:

$$\lambda' L_l(\theta) + (1 - \lambda') L_u(\theta), \tag{7.17}$$

for a sequence of λ', maximize these modified log-likelihood functions to obtain $\hat{\theta}_{\lambda'}$, and choose the best one with respect to cross-validation or testing. The estimator has the same form of Expression (4.9). In fact, asymptotically the estimator is simply modifying the ratio of labeled to unlabeled samples for any fixed λ'. Note that this estimator can only make sense under the assumption that the model is incorrect. Otherwise, both terms in Expression (7.17) lead to unbiased estimators of θ. There is then no reason to impose different weights on the data, and much less reason to search for the best weight, when the differences are solely in the rate of reduction of variance. Presumably, there are a few labeled samples available and a large number of unlabeled samples; why should we increase the importance of the labeled samples, giving them more weight to a term that will contribute more heavily to the variance?

Nigam et al's estimator does seem to have a more solid motivation in the presence of modeling errors. Consider the following scenario, which curiously is not even mentioned by Nigam et al [Nigam et al., 2000]. In Example 4.6 from Chapter 4, if the boundary from labeled data and the boundary from unlabeled data are in different sides of the best linear boundary, then obviously we can find the best linear boundary by changing λ'. We can improve on both supervised and unsupervised learning in such a situation! Some authors have argued that labeled data should be given more weight on somewhat subjective grounds [Corduneanu and Jaakkola, 2002], but this example shows that there are no guarantees concerning the supposedly superior effect of labeled data.

In any case, one cannot expect to find the best possible boundary just by changing λ' in Expression (7.17). Consider the following simple tests with Nigam et al's estimators, taking both artificial and real datasets.

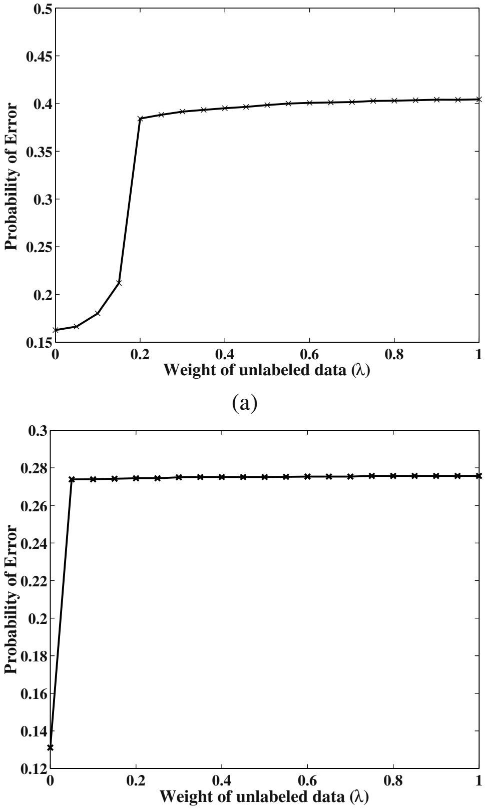
- We first generated data from a randomly generated TAN structure with 10 features, where the Bayes error is approximately 7.51%. We changed the proportion of labeled to unlabeled samples, learning various classifiers with the EM algorithm, obtaining the classification errors depicted in Figure 7.5.

- Consider then the Shuttle dataset from the UCI repository [Blake and Merz, 1998]. Using only 100 labeled samples, a Naive Bayes classifier produced classification error of 18%. Now, with 100 labeled samples and 43400 unlabeled samples, a Naive Bayes learned with the EM algorithm produced classification error of 30%. We then considered various combinations of these labeled and unlabeled datasets. Results are again shown in Figure 7.5.

These tests indicate that Nigam et al's estimators are not uniformly useful in practice.

8. Active Learning

All the methods above considered a ‘passive’ use of unlabeled data. A different approach is known as active learning, in which an oracle is queried



(a)

(b)

Figure 7.5. The effect of varying the weight of the unlabeled data on the error for the Artificial data problem (a) and Shuttle database (b). The number of labeled and unlabeled records is specified in Table 7.3.

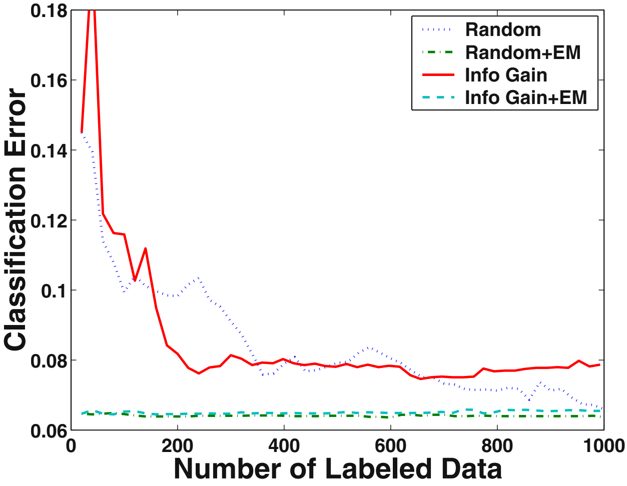
for the label of some of the unlabeled data. Such an approach increases the size of the labeled data set, reduces the classifier's variance, and thus reduces the classification error. There are different ways to choose which unlabeled data to query. The straightforward approach is to choose a sample randomly. This approach ensures that the data distribution $p(C, \mathbf{X})$ is unchanged, a desirable property when estimating generative classifiers. However, the random sample approach typically requires many more samples to achieve the same performance as methods that choose to label data close to the decision boundary. We note that, for generative classifiers, the latter approach changes the data distribution therefore leading to estimation bias. Nevertheless, McCallum and Nigam [McCallum and Nigam, 1998] used active learning with generative models with success. They proposed to first actively query some of the labeled data followed by estimation of the model's parameters with the remainder of the unlabeled data.

We are interested in seeing the effect of active learning for learning a generative Bayesian network. Figure 7.6 brings an interesting example. The figure shows the classification error as a function of the number of (actively queried) labeled data when estimating an arbitrary generative Bayesian network with a Bayes error rate of approximately 6.3%. The classifier includes nine binary observables, with many arcs between the observable nodes, and a binary class variable. We use the simple random active learning approach (labeling of random samples) and the approach of choosing samples with high information gain (i.e., around the decision boundary). We use an initial pool of 20 labeled samples and sample additional samples from a pool of 10000 unlabeled samples. We also test McCallum and Nigam's approach of using the remainder of the unlabeled data after a portion are labeled actively. Figure 7.6(a) shows this approach when the correct structure is specified. We see that for the methods using only labeled data, when we increase the number of labeled data the performance improves at more or less the same rate. We also see that the addition of labeled samples does not make any difference when adding the unlabeled data (Random+EM, Info Gain+EM)— the unlabeled data are already sufficient to achieve an error rate close to the Bayes error. Figure 7.6(b) shows the performance when an incorrect model (TAN) is assumed. Here we see that queries of samples close to the decision boundary allows a faster improvement in performance compared to random sampling — in fact, the bias introduced by the boundary sampling allows the classifier to achieve better overall performance even after 1000 samples. When adding the unlabeled data, performance degradation occurs and as more labeled data are added (changing the ratio between labeled and unlabeled data), the classification performance does improve, but, at 1000 labeled samples, is still far from that using no unlabeled data.

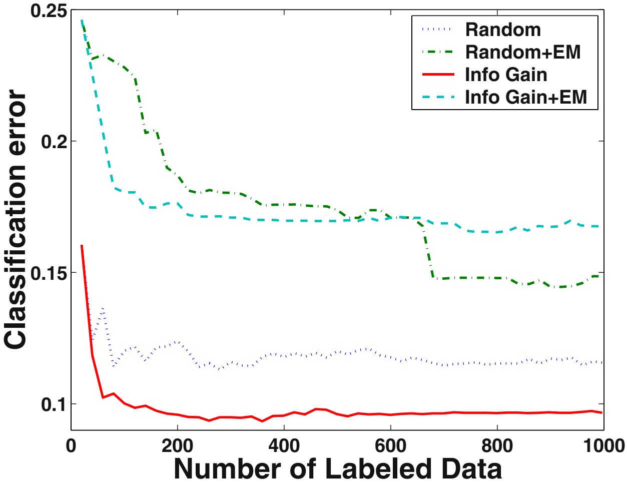
The above results suggest the following statements. With correctly specified generative models and a large pool of unlabeled data, "passive" use of the unlabeled data is typically sufficient to achieve good performance. Active learning can help reduce the chances of numerical errors (improve EM starting point, for example), and help in the estimation of classification error. With incorrectly specified generative models, active learning is very profitable in quickly reducing error, while adding the remainder of unlabeled data might not be desirable.

9. Concluding Remarks

The discussion in this chapter illustrated the importance of structure learning when using unlabeled data for Bayesian network classifiers. We discussed different strategies that can be used, such as model switching and structure learning. We introduced a classification driven structure search algorithm based on



(a)



(b)

Figure 7.6. Learning Generative models with active learning. (a) Assuming correct structure, (b) Assuming incorrect structure (TAN).

Metropolis-Hastings sampling, and showed that it performs well both on fully labeled datasets and on labeled and unlabeled training sets.

The idea of structure search is particularly promising when unlabeled data are present. It seems that simple heuristic methods, such as the solution proposed by Nigam et al [Nigam et al., 2000] of weighing down the unlabeled data, are not the best strategy for unlabeled data. We suggest that structure search, and in particular stochastic structure search, holds the most promise for handling large amount of unlabeled data and relatively scarce labeled data for classification. We also believe that the success of structure search methods for classification increases significantly the breadth of applications of Bayesian networks.

To conclude, this chapter presented and compared several different approaches for learning Bayesian network classifiers that attempt to achieve improvement of classification performance when learning with unlabeled data. While none of these methods is universally superior, each method is useful under different circumstances:

- Learning TANs using EM-TAN is a fast and useful method. Although the expressive power of TAN structures is still limited, it usually outperforms the Naive Bayes classifier in semi-supervised learning.

- Structure learning in general, and classification driven structure search in particular is usually the best approach to learn with unlabeled data. However, this approach comes at a computational cost and would work only with reasonably large datasets.

- It is difficult to learn structure using independence tests and unlabeled data–the reliability of the independence tests is not sufficient.

- Active learning is a promising approach and should be used whenever model uncertainty is prevalent. An interesting idea is to try and use active learning to assist in learning the model, querying data that will assist in determining whether a model change is called for. Pursuit of this idea in the subject of future research.

In a nutshell, when faced with the option of learning with labeled and unlabeled data, our discussion suggests following the following path.

- Start with Naive Bayes and TAN classifiers, learn with only labeled data and test whether the model is correct by learning with the unlabeled data, using EM and EM-TAN.

- If the result is not satisfactory, then SSS can be used to attempt to further improve performance with enough computational resources.

- If none of the methods using the unlabeled data improve performance over the supervised TAN (or Naive Bayes), active learning can be used, as long as there are resources to label some samples.

Chapters 10 and 11 present two different human computer interaction applications (facial expression recognition and face detection) that use the algorithms suggested in this chapter.

Chapter 8

APPLICATION: OFFICE ACTIVITY RECOGNITION

Researchers and application developers have long been interested in the promise of performing automatic and semi-automatic recognition of human behavior from observations. Successful recognition of human behavior is critical in a number of compelling applications, including automated visual surveillance and multimodal human-computer interaction (HCI) – user interfaces that consider multiple streams of information about a user's behavior and the overall *context* of a situation. Although there has certainly been progress on multiple fronts, many challenges remain for developing machinery that can provide rich, human-centric notions of context. Endowing computers with richer notions of context can enhance the communication between humans and computers and catalyze the development of new kinds of computational services and experiences.

In this chapter, we develop probabilistic machinery that can provide real-time interpretations of human activity in and around an office. In brief, we make use of special architecture which we call Layered HMM and our results show that this new proposed model is highly effective in modeling the human activities. We will start by describing the context-sensitive systems.

1. Context-Sensitive Systems

Location and identity have been the most common properties considered as comprising a user's situation in "context-aware" HCI systems. Context can include other critical aspects of a user's situation, such as the user's current and past activities and intentions. Recent work on the use of probabilistic models for reasoning about a user's location, intentions, and focus of attention have highlighted opportunities for building new kinds of applications and services [Horvitz et al., 1999].

Most of the previous work on leveraging perceptual information to recognize human activities has centered on the identification of a specific type of activity in a particular scenario. Many of these techniques are targeted at recognizing single, simple events, e.g., *"waving the hand" or "sitting on a chair"*. Less effort has been applied to research on methods for identifying more complex patterns of human behavior, extending over longer periods of time.

In particular, we would highlight here a number of approaches that employ graphical models for the recognition of human activities. A significant portion of work in this arena has harnessed Hidden Markov Models (HMMs) [Rabiner, 1989]. Starner and Pentland in [Starner et al., 1998] use an HMM for recognizing hand movements used to relay symbols in American Sign Language. The different signs are recognized by computing the probabilities that models for different symbols would have produced the observed visual sequence. More complex models, such as Parameterized-HMM (PHMM) [Wilson and Bobick, 1998], Entropic-HMM [Brand and Kettnaker, 2000], Variable-length HMM (VHMM) [Galata et al., 2001] and Coupled-HMM (CHMM) [Brand et al., 1997], have been used to recognize more complex activities such as the interaction between two people. Bobick and Ivanov [Ivanov and Bobick, 2000], propose the use of a stochastic context-free grammar to compute the probability of a temporally consistent sequence of primitive actions recognized by HMMs. Clarkson and Pentland [Clarkson and Pentland, 1999] model events and scenes from audiovisual information. They have developed a wearable computer system that automatically clusters audio-visual information into events and scenes in a hierarchical manner. Their goal is to determine where the user is at each instant of time (i.e. at home, the office, at the bank, etc). Brand and Kettnaker in [Brand and Kettnaker, 2000] propose an entropic-HMM approach to organize the observed video activities (office activity and outdoor traffic) into meaningful states. They illustrate their models in video monitoring of office activity and outdoor traffic. In [Hongeng et al., 2000], a probabilistic finite-state automaton (a variation of structured HMMs) is used for recognizing different scenarios, such as monitoring pedestrians or cars on a freeway. Although HMMs appear to be robust to changes in the temporal segmentation of observations, they tend to suffer from a lack of structure, an excess of parameters, and an associated overfitting of data when they are applied to reason about long and complex temporal sequences with limited training data. Finally, in recent years, more complex Bayesian networks have also been adopted for the modeling and recognition of human activities [Binder et al., 1997; Madabhushi and Aggarwal, 1999; Hoey, 2001; Horvitz et al., 1998; Fernyhough et al., 1998; Buxton and Gong, 1995; Intille and Bobick, 1999; Horvitz et al., 1999; Forbes et al., 1995].

To date, however, there has been little research on real-time, multimodal systems for human-computer interaction that use probabilistic methods to model typical human activities in a hierarchical manner. The methods and working system described in this chapter focus on this representation. We show how with our approach one can learn and recognize on-the-fly the common, distinct situations in office settings.

2. Towards Tractable and Robust Context Sensing

A key challenge in inferring human-centric notions of context from multiple sensors is the fusion of low-level streams of raw sensor data –for example, acoustic and visual cues– into higher-level assessments of activity. Low-level sensors present a system with relatively high-frequency information. The task of moving from low-level signals that are marked by relatively high-frequency variations to more abstract hypotheses about activity brings into focus a consideration of a spectrum of approaches. Potentially valuable methods include template matching, context-free grammars, and various statistical methods. We have developed a probabilistic representation based on a tiered formulation of dynamic graphical models that we refer to as layered Hidden Markov Models (LHMMs).

To be concrete, let us consider a set of real-world sensors. In particular, we have explored the challenge of fusing information from the following sources:

1. **Binaural microphones:** Two mini-microphones ($20 - 16000$ Hz, SNR 58 dB) capture ambient audio information. The audio signal is sampled at 44100 KHz. The microphones are used for sound classification and localization.

2. **USB camera:** A video signal is obtained via a standard USB camera (Intel), sampled at 30 frames/s. The video input is used to determine the number of persons present in the scene;

3. **Keyboard and mouse:** We keep a history of keyboard and mouse activities during the past 5 seconds.

In our initial work, we attempted to build single-layer (non-hierarchical) models to reason about overall office situation, including determining the presence of a PHONE CONVERSATION, A FACE TO FACE CONVERSATION, A ONGOING PRESENTATION, A DISTANT CONVERSATION, NOBODY IN THE OFFICE and A USER IS PRESENT AND ENGAGED IN SOME OTHER ACTIVITY. Some of these activities have been proposed in the past as indicators of a person's availability [Johnson and Greenberg, 1999].

We explored the use of Hidden Markov Models (HMMs). Hidden Markov models (HMMs) are a popular probabilistic framework for modeling processes that have structure in time. An HMM is essentially a quantization of a system's configuration space into a small number of discrete states, together with probabilities for the transitions between states. A single finite discrete variable indexes the current state of the system. Any information about the history of the process needed for future inferences must be reflected in the current value of this state variable. There are efficient algorithms for state and parameter estimation in HMMs. Graphically HMMs are often depicted “rolled-out in time”, such as in Figure 8.1 (a).

We found that a single-layer HMM approach generated a large parameter space, requiring substantial amounts of training data for a particular office or user, and with typical classification accuracies not high enough for a real application. Finally and more importantly, when the system was moved to a new office, copious retraining was typically necessary to adapt the model to the specifics of the signals and/or user in the new setting.

Therefore, we sought a representation that would be robust to typical variations within office environments, such as changes of lighting and acoustics. We wanted a representation that would allow the models to perform well when transferred to new office spaces with minimal tuning through retraining. We also pursued a representation that would map naturally onto the problem space. Psychologists have found that many human behaviors are hierarchically structured [Zacks and Tversky, 2001]. We desired a representation that could capture such hierarchical properties in an elegant manner.

We converged on the use of a multilevel representation that allows for explanations at multiple temporal granularities, by capturing different levels of temporal detail. For example, in the domain of office awareness, one level of description is the analysis and classification of the raw sensor signals (e.g., audio, keyboard, mouse, video, etc.). This level corresponds to a fine time granularity of the order of milliseconds. Another level of description is the detection of the user’s presence. In this case, the time granularity is of the order of a second. At another level, one could describe what the user has done in the last N minutes. Finally, one could provide an explanation of the user’s activities during the day.

3. Layered Hidden Markov Models (LHMMs)

We have developed a layered HMM (LHMM) representation in an attempt to decompose the parameter space in a way that could enhance the robustness of the system by reducing training and tuning requirements. In LHMMs, each layer of the architecture is connected to the next layer via its inferential results. The representation segments the problem into distinct layers that operate at different temporal granularities – allowing for temporal abstractions from point-wise observations at particular times into explanations over varying temporal intervals. LHMMs can be regarded as a cascade of HMMs. The structure of a three-layer LHMM is displayed in Figure 8.1 (b).

3.1 Approaches

Formally, given a set of T_L observations, $O^L = \{O_1^L, O_2^L, ..., O_{T_L}^L\}$, at level L, the HMMs at this level, can be thought of as a multiclass classifier mapping these observations with time granularity T_L to one of K_L classes. If $O_t^L \in \mathcal{X}^{T_L}$, then the bank of HMMs can be represented as $f : \mathcal{X}^{T_L} \rightarrow \mathcal{Y}^L$ where $\mathcal{Y}^L = \{0, 1, ..., K_L - 1\}$. The HMMs at the next level $(L + 1)$ take as inputs the outputs of the HMMs at level L and learn a new classification function with time granularity T_{L+1}, $f : \mathcal{X}^{T_{L+1}} \rightarrow \mathcal{Y}^{L+1}$, where $\mathcal{Y}_t^L \in \mathcal{X}^{T_{L+1}}$. In this framework, each HMM is learned independently of the others. The availability of labeled data during the training phase allows us to do efficient supervised learning. By itself, each HMM is trained using the Baum-Welch algorithm [Rabiner, 1989].

A layered HMM structure provides valuable properties. Layered formulation makes it feasible to decouple different levels of analysis for training and inference. As we review in Section 8.4.3, each level of the hierarchy is trained independently, with different feature vectors and time granularities. In consequence, the lowest, signal-analysis layer that is most sensitive to variations in the environment could be retrained, while leaving the higher-level layers unchanged.

We have implemented two approaches to do inference with LHMMs. In the first approach, which we refer to as *maxbelief*, models with the highest likelihood are selected, and this information is made available as an input to the HMMs at the next level. In the *distributional* approach, we execute the low-level models and pass a probability distribution over the models to the higher-level HMMs.

As an example, let us suppose that we train K HMMs at level L of the hierarchy, $M(k)^L$, with $k = 1, ..., K$. Let

$$\mathcal{L}(k,t)^L = \log(P(O(1:t)|M(k)^L)) = \log \sum_i \alpha_t(i; M(k)^L)$$

be the log-likelihood of model $M(k)^L$ given all the observations up to time t; and let $\alpha_t(i; M(k)^L)$ be the alpha variable at time t for model $M(k)^L$.

At that level, we classify the observations by declaring

$$C(t)^L = \arg\max_k \mathcal{L}(k)_t^L,$$

with $k = 0, ..., K - 1$. The next level of the hierarchy $(L + 1)$ could have two kinds of observations of τ temporal length:

1 either, $C(1 : \tau)^L$, *i.e.* the hard classification results from the previous level for each time step and therefore a vector of τ discrete symbols in $\{0, ..., K - 1\}$. This is the *maxbelief* approach, or

2 $\{\mathcal{L}(0:K-1)_{t=1}^{L}, ..., \mathcal{L}(0:K-1)_{t=\tau}^{L}\}$, *i.e.* the log-likelihoods for each of the models and time instants, –and therefore a vector of K reals for each time step. This corresponds to the *distributional* approach.

In our experience, we did n0t observe any better performance by using the latter approach. The results reported in section 8.5 correspond to the *maxbelief* approach, which is simpler.

3.2 Decomposition per Temporal Granularity

Figure 8.1 (b) highlights how we decompose the problem into layers with increasing time granularity. For example, at layer L we have a sliding time window of T^L samples. The HMMs at this level analyze the data contained in such time window, compute their likelihood and generate one observation for layer $L+1$ for each processed time window. That observation is the inferential output of the HMMs in level L. The sliding factor along with the window length vary with the granularity that is represented at each level. At the lowest level of the hierarchy, the samples of the time window are the features extracted from the raw sensor data (see Section 8.4.1). At any other level of the hierarchy, the samples are the inferential outputs of the previous level. The higher the level, the larger the time scale – and therefore the higher the level of abstraction – because gathering observations at a higher level requires the outputs of lower layers. In a sense, each layer performs time compression before passing data upward.

Automatic estimation of T^L from data is a challenging problem both for standard HMMs and LHMMs. In the models and experiments described in this chapter, we determined the time granularities at each level based on our intuitions and knowledge about the different classes being modeled at each level. We used cross-validation to select the optimal values from the original set of proposed ones.

Focusing more specifically on our target application of office awareness, we employ a two layer HMM architecture. The raw sensor signals are processed with time windows of duration less than 100 milliseconds. Next, the lowest layer of HMMs classify the audio and video data with a time granularity of less than 1 second. The second layer of HMMs represents typical office activities, associated with a time granularity of about $5-10$ seconds. The activities modeled in this setting are: (1) PHONE CONVERSATION; (2) PRESENTATION; (3) FACE-TO-FACE CONVERSATION; (4) USER PRESENT, ENGAGED IN SOME OTHER ACTIVITY; (5) DISTANT CONVERSATION (outside the field of view); (6) NOBODY PRESENT.

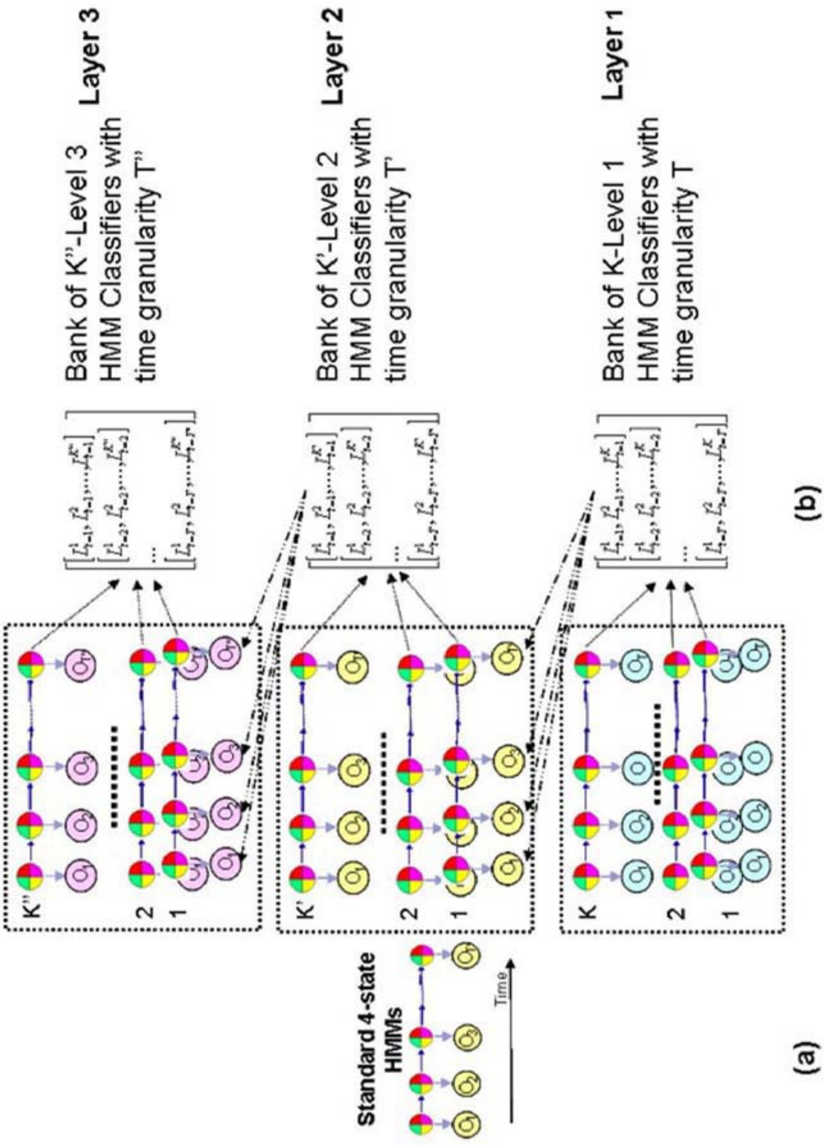

Layer 3
Bank of K''-Level 3 HMM Classifiers with time granularity T''
Layer 2
Bank of K'-Level 2 HMM Classifiers with time granularity T'
Layer 1
Bank of K-Level 1 HMM Classifiers with time granularity T
K''
2
1
K'
K
Standard 4-state HMMs
Time
(a)
(b)

Figure 8.1. Graphical representation of (a) HMMs and rolled-out in time, and (b) an architecture of layered HMMs with 3 different levels of temporal granularity.

4. Implementation of SEER

We explored the use of LHMMs in a system named SEER, which employs a two-layer HMM architecture.

4.1 Feature Extraction and Selection in SEER

The raw sensor signals are preprocessed to obtain feature vectors (*i.e.* observations) for the first layer of HMMs. With respect to the audio analysis, Linear Predictive Coding coefficients [Rabiner and Huang, 1993] are computed. Feature selection is applied to these coefficients via principal component analysis. The number of features is selected such that at least 95% of the variability in the data is maintained, which is typically achieved with no more than 7 features. We also extract other higher-level features from the audio signal such as its energy, the mean, and variance of the fundamental frequency over a time window, and the zero crossing rate [Rabiner and Huang, 1993], given by

$$Zero_s(m) = \frac{1}{N} \sum_{n=m-N+1}^{m} \frac{|sign(s(n)) - sign(s(n-1))|}{2} \cdot w(m-n),$$

where m is the frame number, N is the frame length, w is a window function, $s(n)$ is the digitized audio signal at an index indicator n, and

$$sign(s(n)) = \begin{cases} 1 & \text{if } s(n) \geq 0 \\ -1 & \text{if } s(n) < 0. \end{cases}$$

The source of the sound is localized using the Time Delay of Arrival (TDOA) method. In TDOA [Brandstein and Silverman, 1997], one measures the time delays between the signals coming from each sensor. Typically, TDOA-based approaches have two steps: the time delay estimation and the sound source localization. Let $s(n)$ be the source signal and let $x_i(n)$ be the i-th sensor received signal. If we assume no reverberation, we have $x_i(n) = a_i s(n - t_i) + b_i(n)$. To model reverberation, one can add a non-linear reverberation function: $x_i(n) = g_i * s(n - t_i) + b_i(n)$, where a_i is the attenuation factor, b_i is additive noise and g_i is the response between the source and the sensor.

In SEER we implemented multiple approaches for estimating the time delay of arrival between the left and right audio signals. We obtained the best performance by estimating the peak of the time cross-correlation function between the left and right audio signals over a finite time window $[N_1, N_2]$, *i.e.*:

$$r_{lr}(d) = \sum_{n=N_1}^{n=N_2} l(n) r(n-d).$$

This is the method used in the experiments described in Section 8.5.

With respect to the video, four features are extracted from the video signal: the density of skin color in the image (obtained by discriminating between skin and non-skin models, consisting of histograms in HSV color space), the density of motion in the image (obtained by image differences), the density of foreground pixels in the image (obtained by background subtraction, after having learned the background), and the density of face pixels in the image (obtained by applying a real-time face detector to the image [Li et al., 2001]).

Finally, a history of the last 5 seconds of mouse and keyboard activities is logged.

4.2 Architecture of SEER

We employ a two-level cascade of HMMs. The lowest level captures video, audio, and keyboard and mouse activity, and computes the feature vectors associated to each of these signals, as previously described.

The middle layer includes two banks of distinct HMMs for classifying the audio and video feature vectors. The structure for each of these HMMs is determined by means of cross-validation on a validation set of real-time data. On the audio side, we train one HMM for each of the following office sounds: human speech, music, silence, ambient noise, a phone ringing, and the sounds of keyboard typing. We will denote this kind of HMMs “discriminative HMMs”. When classifying the sounds, all of the models are executed in parallel. At each instant, the model with the highest likelihood is selected and the sound is classified correspondingly. The source of the sound is also localized using a technique based on the Time Delay of Arrival (TDOA), as explained before. The video signals are classified using another set of discriminative HMMs that implement a person detector. At this level, the system detects whether one person is present in the room (semi-static), one active person is present, multiple people are present, or there is nobody in the office.

The inferential results[1] from this layer (i.e., the outputs of the audio and video classifiers), the derivative of the sound localization component, and the history of keyboard and mouse activities constitute a feature vector that is passed to the next (third) and highest layer of analysis. This third layer handles concepts that have longer temporal extent. Such concepts include the user’s typical activities in or near an office. The models at this level are also discriminative HMMs.

[1] See section 8.3 for a detailed description of how we use these inferential results.

4.3 Learning in SEER

Since in SEER, at each layer of the hierarchy, each classifier is a HMM, learning is reduced to the problem of learning individual HMMs. This problem is typically solved by making use of the forward-backward or Baum-Welch algorithm. This algorithm provides expressions for the forward, $\alpha_t(i)$, and backward, $\beta_t(i)$, variables, as

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) P_{j|i}\right] p_j(o_t) \tag{8.1}$$

and for the $\beta_t(i)$ variable,

$$\beta_t(i) = \left[\sum_{j=1}^{N} \beta_{t+1}(j) P_{i|j} p_j(o_{t+1})\right] \tag{8.2}$$

where N is the number of hidden states, P is the transition probability matrix with $P_{i|j}$ being the probability of state i given state j, and $p_i(o_t)$ is the probability for observation at time t given that the model is in state i. From the α and β variables one can obtain the model parameters, *i.e.* the observation and transition probabilities. These can also be used to obtain the log-likelihood of a sequence of observations which is given by

$$\mathcal{L} = \log P(O_1, ..., O_T) = \log \sum_{i=1}^{N} \alpha_t(i)\beta_t(i).$$

Some more details on this and few extensions of this are described in Chapter 9.

4.4 Classification in SEER

The final goal of the system is to decompose in real-time the temporal sequence obtained from the sensors into concepts at different levels of abstraction or temporal granularity. During the training phase, multiple classifiers at different levels have been learned. Since the classifiers at each level are a set of HMMs, we adopt standard HMM inferencing techniques. We use the forward-backward algorithm to compute the likelihood of a sequence given a particular model at a particular level.

5. Experiments

As we have previously mentioned, layered representations are supported by psychological findings that both the production and interpretation of human activities is hierarchical in nature. We also believed that layered representations could provide a means for decomposing the overall context-assessment challenge into a set of simpler subproblems that could make learning and inference more tractable and robust to variation in different instances of the target domain.

We have tested SEER in multiple offices, with different users, and respective environments for several weeks. In our tests, we have found that the high-level layers of SEER are relatively robust to changes in the environment.

In all the cases, when we moved SEER from one office to another, we obtained nearly perfect performance *without* the need for retraining the higher levels of the hierarchy. Only some of the lowest-level models required retraining to tune their parameters to the new conditions (such as different ambient noise, background image, and illumination) . The fundamental decomposability of the learning and inference of LHMMs makes it possible to reuse prior training of the higher-level models, allowing for the selective retraining of layers that are less robust to the variations present in different instances of similar environments.

In a more quantitative study, we compared the performance of our model with that of single, standard HMMs. The feature vector in the latter case results from the concatenation of the audio, video, and keyboard/mouse activities features in one long feature vector. We refer to these HMMs as the Cartesian Product (CP) HMMs. For example, in SEER we want to classify 6 different high-level office activities. Let us assume that we use eight-state CP HMMs with single Gaussian observations of dimensionality 16 to model such behaviors. We would need to estimate $8 * (16 + 16 + 120) = 1216$ parameters for each behavior. An equivalent LHMM with 2 levels would typically have, at the lowest level, two banks of, say, five-state HMMs (six audio HMMs – assuming we have six audio classes, and four video HMMs – assuming four video classes, with dimensionalities 10 and 3 respectively), and at the highest level (the behavior level), a set of 6 four-state HMMs[2] of dimensionality 12, if we use the *distributional* approach: six dimensions for the six audio HMMs, four for the four video HMMs, one for the sound localization component, and another for the keyboard/mouse activities. This amounts to $5 * 5 * (10 + 10 + 45) + 4 * 5 * (3 + 3 + 3) = 1805$ for all the models at the first layer and $4 * (12 + 12 + 66) = 360$ for each behavior at the second layer. Therefore, the number of parameters to estimate is lower for LHMMs than for the CP HMMs. Moreover, the inputs at each level have already been filtered by the previous level and are more stable than the feature vectors directly extracted from the raw sensor data. In summary, encoding prior knowledge about the problem in the structure of the models decomposes the problem in simpler subproblems and reduces the dimensionality of the overall model. Therefore,

[2] In our experiments the best models obtained using cross-validation had no more than 4 states in LHMMs but needed at least 8 states in the Cartesian Product HMMs.

for the same amount of training data, it is expected for LHMMs to have superior performance than HMMs. Our experimental results corroborate such expectation. We would like to highlight at this point that it is not considerably more difficult to determine the structure of LHMMs when compared to HMMs. Both for HMMs and LHMMs we estimated the structure of each of the models – and at each of the levels for LHMMs – using cross-validation. The only additional complexity when designing an LHMM architecture is the number of levels and their time granularity. In our experience, however, we found that our intuition and knowledge about the problem were enough to estimate how many layers to use and the time span of each layer.

Figure 8.2 illustrates the per-frame normalized likelihoods on testing in *real-time* both HMMs and LHMMs with the different office activities. By "normalized" likelihoods, we denote the likelihoods whose values have been bounded between 0 and 1. They are given by:

$$NormL_i = \frac{\mathcal{L}_i - min_j(\mathcal{L}_j)}{max_j(\mathcal{L}_j) - min_j(\mathcal{L}_j)},$$

for $i = 1, ..., N$, $j = 1, ..., N$, and N models.

Note that, in the case of LHMMs, the likelihoods are those corresponding to the highest level in the hierarchy, because this is the level that models the office activities.

Finally, we carried out a different set of experiments. We trained and tested the performance of LHMMs and HMMs on 60 minutes of recorded office activity data (10 minutes per activity and 6 activities). Given that it was recorded activity data, we knew the ground truth for each activity. The first few seconds of each dataset were ignored for classification purposes, due to the lag of the models in recognizing each activity. We used 50% of the data (i.e 5 minutes per activity) for training. In particular, we used about 20 sequences of each class for the audio and video HMMs (first layer) and 10 sequences of each office activity for the behavior HMMs (second layer). The rest of the data (i.e., 5 min per activity) was used for testing. The results are summarized in Table 8.1. The average accuracies of both HMMs and LHMMs on testing data were of 72.68% (STD 8.15) and 99.7% (STD 0.95), respectively. In our experience with the system, HMMs normally need more training under similar office conditions (lighting, acoustics, etc.) than that of the particular testing data to obtain reasonable classification results. On the other hand, we can normally reuse at least the highest level in LHMMs that have been trained under different office conditions than that of testing.

Table 8.1. Confusion matrix for tuned CP HMMs and generic LHMMs on 30 min of real data, where PC=Phone Conversation; FFC=Face to Face Conversation; P=Presentation; O=Other Activity; NA=Nobody Around; DC=Distant Conversation.

Confusion Matrix for tuned CP HMMs

	PC	FFC	P	O	NA	DC
PC	0.8145	0.0679	0.0676	0.0	0.0	0.05
FFC	0.0014	0.9986	0.0	0.0	0.0	0.0
P	0.0	0.0052	0.9948	0.0	0.0	0.0
O	0.0345	0.0041	0.003	0.9610	0.0	0.0
NA	0.0341	0.0038	0.0010	0.2524	0.7086	0.0
DC	0.0076	0.0059	0.0065	0.0	0.0	0.98

Confusion Matrix for generic LHMMs

	PC	FFC	P	O	NA	DC
PC	1.0	0.0	0.0	0.0	0.0	0.0
FFC	0.0	1.0	0.0	0.0	0.0	0.0
P	0.0	0.0	1.0	0.0	0.0	0.0
O	0.0	0.0	0.0	1.0	0.0	0.0
NA	0.0	0.0	0.0	0.0	1.0	0.0
DC	0.0	0.0	0.0	0.0	0.0034	0.9966

5.1 Discussion

Some observations that can be drawn from our experiments are:

1. *For the same amount of training data, the accuracy of LHMMs is significantly higher than that of HMMs.* The number of parameters of the CP HMMs is higher than that of LHMMs for the office activities being modeled in our experiments. As a consequence, for the same limited amount of training data, HMMs are more prone to overfitting and have worse generalization than LHMMs.

2. *LHMMs are more robust to changes in the environment than HMMs.* In our experiments, the HMMs were more sensitive to changes in the environment than LHMMs. We could not obtain any reasonable performance on the CP HMMs had they not been *tuned* to the particular testing environment and conditions. We had to retrain the CP HMMs every time we needed to test them under some particular conditions. On the contrary, at least the highest layer of our LHMMs did *not* require retraining, despite the changes in office conditions. This is due to the fact that the CP HMMs carry out high-level inferences about the user's activity, directly from the raw sensor signals, whereas LHMMs isolate the sensor signals in different sub-HMM models for each input modality. Due to its layered structure, LHMMs are expected

to be more robust to noise in the sensor signals and at the same time to have higher accuracy than the CP HMMs.

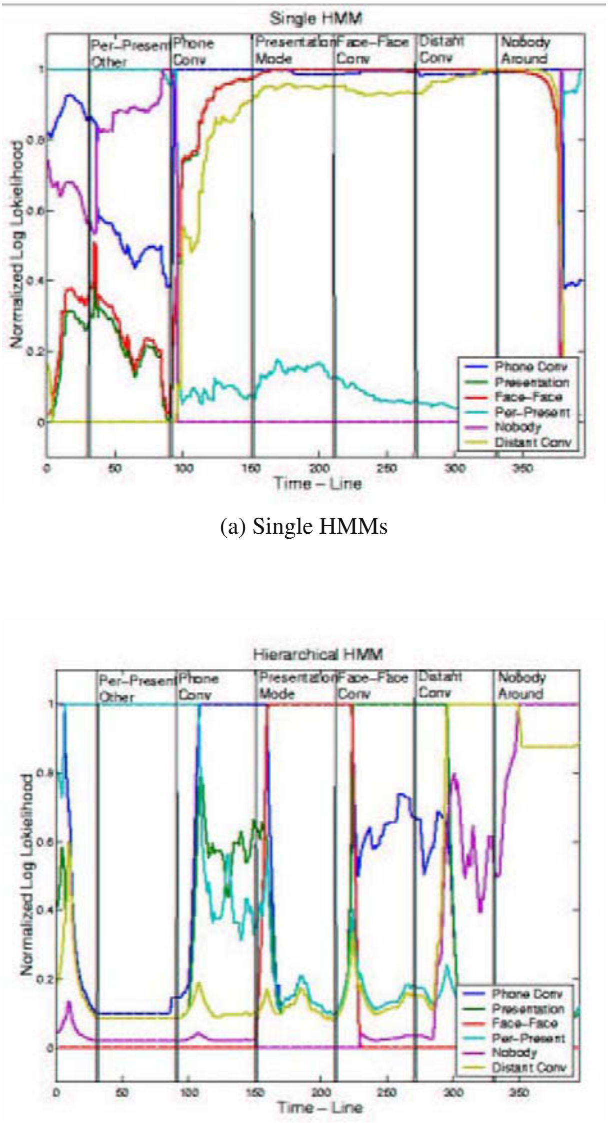
3. *The discriminative power of LHMMs is notably higher than that of HMMs.* By *discriminative power*, we mean the distance between the log-likelihoods of the two most likely models. The log-likelihoods for the CP HMMs tend to be much closer to each other, making them prone to instability and errors in the classification. Note in Figure 8.2 how the normalized likelihoods between the two best models in CP HMMs are much closer than that in LHMMs. This phenomenon is particularly noticeable in the PRESENTATION, FACE TO FACE CONVERSATION, DISTANT CONVERSATION, and NOBODY AROUND activities.

6. Related Representations

HMMs and their extensions, *e.g.*, CHMMs, PHMMs, VHMMs, including the architecture proposed in this chapter, LHMMs, are particular cases of temporal or dynamic graphical models (DGMs). Given suitable independence relationships among the variables over time, DGMs can provide a computationally efficient and sufficiently expressive solution to the problem of human activity modeling and recognition. DGMs have several properties that are useful for human behavior modeling: they can handle incomplete data as well as uncertainty; they are trainable; they encode causality (conditional independency) in a natural way; algorithms exist for doing predictive inference; they offer a framework for combining prior knowledge and data; and finally they are modular and parallelizable.

A formulation for Hierarchical HMMs was first proposed in [Fine et al., 1998] in work that extended the standard Baum-Welch procedure and presented an estimation procedure of the model parameters from unlabeled data. A trained model was applied to an automatic hierarchical parsing of an observation sequence as a dendrogram. Because of the computational complexity of the original algorithm, the authors suggest an efficient approximation to the full estimation scheme. The approximation could further be used to construct models that adapt both their topology and parameters. The authors briefly illustrate the performance of their models on natural written English text interpretation and in English handwriting recognition. Recently, Murphy and Paskin [Murphy and Paskin, 2001] introduced a linear-time inference algorithm for HHMMs.

Hoey [Hoey, 2001] proposes the use of a hierarchical framework for event detection. Although being a nice framework it does not seem to be particularly suited for a task with real-time constraints, because it requires the manual segmentation of the audio/video streams. A new architecture of HMMs called *embedded HMMs* is proposed in [Nefian and Hayes, 1999a]. Such HMMs are



(a) Single HMMs

(b) LHMMs

Figure 8.2. Log-likelihoods for each of the activity models over time when tested in real time

tuned towards applications that handle two-dimensional data – and particularly images. One HMM models one dimension of the data while its state variables correspond to the other dimension of the data. They have successfully applied these to the task of face recognition.

In the original formulation of [Fine et al., 1998] and other related papers ([Hoey, 2001; Murphy and Paskin, 2001]), each state of the architecture is another HMM or variation, and therefore represents a time sequence of the raw signals. In our model, however, at any given level of the hierarchy, there are multiple HMMs each corresponding to a certain concept (for example, we have six HMMs corresponding to different classes of audio signals - speech, silence, music, etc). These HMMs take as observations either the features computed from the raw signals – at the lowest level – or the likelihoods coming from the previous level – at any other level.

The LHMM approach is most closely related to the concept of Stacked Generalization [Wolpert, 1992], where the main idea is to learn classifiers on top of classifiers. Stacked Generalization is a technique proposed to use learning at multiple levels. A learning algorithm is used to determine how the outputs of the base classifiers should be combined. For example, in a two-layer stacked classifier, the original dataset constitutes the "level zero" data. All the base classifiers run at this level. The "level one" data are the outputs of the base classifiers. Another learning process occurs using as input the "level one" data and as output the final classification results. This is a more sophisticated technique than cross-validation and has been shown to reduce the classification error due to the bias in the classifiers. Note that, while HMMs are generative probabilistic models, they can also be treated as classifiers. From this perspective, we can describe our layered HMMs as a representation for learning different stacked classifiers and using them to do the classification of temporal concepts with different time granularities. Rather than training the models at all the levels at the same time, the parameters of the HMMs at each level can be trained independently (provided that the previous level has been already trained), in a bottom-up fashion, by means of the traditional Baum-Welch algorithm. The inputs (observations) of each level are the classification outputs of the previous level, such that only at the lowest level the observations (the leaves of the tree) are the feature vectors extracted directly from sensor signals.

7. Summary

We have described the principles and implementation of a real-time, multimodal approach to human activity recognition in an office environment. We have introduced a layered HMM representation (LHMM) that has the ability to capture different levels of abstraction and corresponding time granularities. The representation and associated inference procedure appear to be well matched to the decomposition of signals and hypotheses for discriminating a set of activities in an office setting. Our models are learned from data and can be trained on-the-fly by the user. Some important characteristics of LHMMs when compared to HMMs are:

1 LHMMs can encode the hierarchical temporal structure of the office activity modeling problem;

2 LHMMs, due to their layered structure, are easier to interpret, and, thus, easier to refine and improve, than the corresponding CP HMMs;

3 the dimensionality of the state space that needs to be learned from data is smaller in LHMMs than that of their corresponding CP HMMs; in consequence, for the same amount of training data, LHMMs are less prone to overfitting than HMMs;

4 LHMMs can encode different levels of abstraction and time granularities that can be linked to different levels of representation for human behaviors;

5 the modularity of LHMMs allows the selective retraining of the levels that are most sensitive to environmental or sensor variation, minimizing the burden of training during transfer among different environments.

We have tested the performance of LHMMs in SEER, a real-time system for recognizing typical office activities. SEER can accurately recognize when a user is engaged in a phone conversation, giving a presentation, involved in a face-to-face conversation, doing some other work in the office, or when a distant conversation is occurring in the corridor. We believe that our framework can be used to enhance multimodal solutions on the path to more natural human-computer interaction.

There are two kind of theoretical issues that need to be addressed. First one deals with refining the probabilistic model that we have used. This involves understanding the influence of the layered decomposition on the size of the parameter space, and the resulting effects on learning requirements and accuracy of inference for different amounts of training. Alternate decompositions lead to layers of different configurations and structure; we are interested in understanding better how to optimize the decompositions. We are also working on comparing our LHMMs representation to other hierarchical representations, and on exploring the use of unsupervised and semi-supervised methods for training one or more layers of the LHMMs without explicit training effort. Finally, we are exploring several applications of inference about context.

However, the even more important issue deals with understanding these classifiers that are being used. This is important in developing better classifiers and solving more real world problems. This issue was extensively addressed in Chapters 2 and 3.

Chapter 9

APPLICATION: MULTIMODAL EVENT DETECTION

In Chapter 8, we have seen how probabilistic models can be used to develop classifiers that can do inference of high level human activities in an office environment. In this chapter, we will look into another application: multimodal event detection.

Detecting semantic events from audio-visual data with spatio-temporal support is a challenging multimedia understanding problem. The difficulty lies in the gap that exists between low-level features and high-level semantic labels. Often, one needs to depend on multiple modalities to interpret the semantics reliably. This necessitates efficient schemes, which can capture the characteristics of high level semantic events by fusing the information extracted from multiple modalities.

Research in fusing multiple modalities for detection and recognition has attracted considerable attention. Most techniques for fusing features from multiple modalities, having temporal support are based on Markov models. Examples include the Hidden Markov Model (HMM) [Rabiner, 1989] and several variants of the HMM, like the Coupled-HMM (CHMM) [Brand et al., 1997], Factorial-HMM [Ghahramani and Jordan, 1997], the Hierarchical HMM [Naphade et al., 1998], etc. A characteristic of these models is the stage, at which the features from the different modalities are merged.

We present a novel algorithm, which combines feature with temporal support from multiple modalities. Two main features that distinguish our model from existing schemes are:

1 the ability to account for non-exponential duration, and

2 the ability to map discrete state input sequences to decision sequences.

The standard algorithms modeling the video-events use HMMs which models the duration of events as an exponentially decaying distribution. However, we argue that the duration is an important characteristic of each event and we demonstrate it by the improved performance over standard HMMs. We test the model on the audio-visual event *explosion*. Using a set of hand-labeled video data, we compare the performance of our model with and without the explicit model for duration. We also compare performance of the proposed model with the traditional HMM and observe that the detection performance can be improved.

1. Fusion Models: A Review

Audio-visual analysis to detect the semantic concepts in videos poses a challenging problem. One main difficulty arises from the fact that the different sensors are noisy in terms of the information they contain about different semantic concepts. For example, based on pure vision, its hard to make out between a explosion and normal fire. Similarly, audio alone may give confusing information. On one hand one may be able to filter out the disambiguity arising from one source of information by looking (analyzing) other source. While, on the other hand, these different source may provide complimentary information which may be essential in inference.

Motivated by these difficulties, in the past few years a lot of research has gone into developing algorithms for fusing information from different modalities. Since different modalities may not be sampled at the same temporal rate, it becomes a challenging problem to seamlessly integrate different modalities (e.g. audio is normally sampled at 44KHz whereas video is sampled at 30 frames/s). At the same time, one may not even have the synchronized streams (sources of information) or the sources of information may have very different characteristics (audio - continuous, inputs to the computer through keyboard - discrete).

If we assume that one can get features from the different streams on a common scale of time, the two main categories of fusion models are those that favor early integration of features versus those that favor late integration. Early integration refers to combining the information at the level of raw features. Simple early integration is often observed in the form of concatenation of weighted features from different streams. More involved models of early integration have been proposed by using some form of Markov models. Brand et al. [Brand et al., 1997] have proposed the coupled hidden Markov models and used it for detection of human activities. Ghahramani et al. [Ghahramani and Jordan, 1997] have proposed the factorial hidden Markov models. The main difference in these models arises from the conditional independence assumptions that they make between the states of the different information sources. They assume that the different sources are tightly coupled and model them using a single generative process.

In many situations, especially when the different sources are providing complimentary information one may prefer late integration. This refers to doing inferencing of each stream independently of the others and then combining the output of the two. This is especially important and shows improved results as now one essentially looks at the essential information contained in the different streams and the sensor dependent characteristics do not play any role. It also allows one to learn different models for each source independently of one another and then combine the output. One may simply look at the weighted decisions of different sources or may actually use probabilistic models to model the dependencies. For example, Garg et al. [Garg et al., 2000b; Garg et al., 2003] have proposed the use of dynamic Bayesian networks over the output of the different streams to solve the problem of speaker detection. Similarly, Naphade et al. [Naphade et al., 1998] have proposed the use of hierarchical HMMs for solving the problem of event detection. Another powerful model to solve the of activity recognition is LHMMs, as was seen in the Chapter 8.

We observed that in the case of movie, the audio and the visual streams normally carry complimentary information. For example, a scene of explosion is not just characterized by a huge thunder but also a visual effect corresponding to bright red and yellow colors. Motivated by this fact we propose the use of late coupling which seems to suit better for this framework.

Fusion of multimodal feature streams (especially audio and visual feature streams) has been applied to problems like bimodal speech [Chen and Rao, 1998], multimodal speaker detection [Garg et al., 2000b], summarization of video [Nakamura and Kanade, 1997], query by audio-visual content [Naphade et al., 2001], and event detection in movies [Naphade et al., 1998]. Examples of fusion of other streams include fusion of text and image content, motion, and image content etc. In Chapter 8, we saw the fusion of audio, video, along with the computer activity.

2. A Hierarchical Fusion Model

This chapter discusses the problem of fusing multiple feature streams enjoying spatio-temporal support in different modalities. We present a hierarchical fusion model (see Figure 9.1), which makes use of late integration of intermediate decisions.

To solve the problem we propose a new fusion model: the Hierarchical Input output duration dependent Markov model. There are three main considerations that have led us to the particular choice of the fusion architecture.

- We argue that, the different streams contain information which is correlated to another stream only at high level. This assumption allows us to process output of each source independently of one another. Since these sources may contain information which has highly temporal structure, we propose the use of hidden Markov models. These models are learned from the data and then we decode the hidden state sequence, characterizing the information describing the source at any given time.

- We argue that the different sources contain independent information (this assumption may not be true in general and the effect of this on the classification error was analyzed in Chapter 2). At high level, the output of one source is essentially independent of the information contained in the other. However, conditioned upon a particular bi-modal concept, these different sources may be dependent on one another. This suggests the use of input output Markov model. These models are able to capture the correlation that is present between the different sources. These are discussed in detail in the next section. In simple terms, these can be thought of HMMs with observation dependent transition probability matrix.

- Finally, an important characteristic of semantic concepts in videos is their duration. This points to the important limitation of hidden Markov models. In HMMs the probability of staying in any particular state decays exponentially. This is the direct outcome of the one Markov property of these models. To alleviate this problem, we explicitly model these probabilities. This leads us to what we call duration dependent input output Markov model. Note that because of this explicit modeling of the state duration, these are not really Markov models but are what have been called in past semi-Markov models.

2.1 Working of the Model

Consider S streams of features. For each stream, consider feature vectors $f_1^{(s)}, \ldots, f_T^{(s)}$ corresponding to time instants t=$1, \ldots, T$. Consider two hypotheses H_i, $i \in \{0, 1\}$ corresponding to the presence and absence of an event E in the feature stream. Under each hypothesis we assume that the feature stream is generated by a hidden Markov model [Rabiner, 1989]. We estimate the parameters of the HMM under each hypothesis, by using the EM algorithm [Rabiner, 1989] and a labeled training set. Having estimated the parameters of the models under both hypotheses, we then evaluate the best state sequence and hypothesis for the test set through maximum-likelihood detection. Once the state sequence for each feature stream is obtained, we can use these intermediate-level decisions from that feature stream [Naphade et al., 1998] in a hierarchical approach.

Figure 9.1 shows two streams one referring to audio features or observations $AO_1, ...AO_T$ and the other to video features $VO_1, \ldots, VO_T$. The media features act as observations for the media HMMs. The HMMs decode the hidden state sequence $A_1, \ldots, A_T$ and $V_1, \ldots, V_T$. These state sequences form the input to our top level DDIOMM, which then predicts high level semantic information (as a decision sequence $Q_1, \ldots, Q_T$).

Each state in the media HMMs represents a stationary distribution and by using the Viterbi decoder over each feature stream, we essentially cluster features spatio-temporally and quantize them through state identities. State-sequence-based processing using trained HMMs can thus be thought of as a form of guided spatio-temporal vector quantization for reducing the dimensionality of the feature vectors from multiple streams [Naphade et al., 1998]. In the next section we present a model for fusing the state-sequence based features from multiple modalities and an algorithm for estimating the best decision sequence.

2.2 The Duration Dependent Input Output Markov Model

Consider a sequence $\mathcal{Y}$ of symbols $y_1, \ldots, y_T$ where $y_i \in \{1, \ldots, N\}$. Consider another sequence $\mathcal{Q}$ of symbols $q_1, \ldots, q_T$, $q_i \in \{1, \ldots, M\}$. The model between the dotted lines in Figure 9.1 illustrates a Bayesian network involving these two sequences. Here $y = \{A, V\}$ (i.e. Cartesian product of the audio and video sequence) . This network can be thought of as a mapping of the input sequence $\mathcal{Y}$ to the output sequence $\mathcal{Q}$. We term this network the *Input Output Markov Model.* This network is close in spirit to the input output hidden Markov model [Bengio and Frasconi, 1996].

The transition in the output sequence is initially assumed to be Markovian. This leads to an exponential duration density model. Let us define $A = \{A_{ijk}\}$, $i, j \in \{1, \ldots, M\}$, $k \in \{1, \ldots, N\}$ as the map. $A_{ijk} = P(q_t = j | q_{t-1} = i, y_t = k)$ is estimated from the training data through frequency counting and tells us the probability of the current decision state given the current input symbol and the previous state. Once A is estimated, we can then predict the output sequence $\mathcal{Q}$ given the input sequence $\mathcal{Y}$ using Viterbi decoding.

The algorithm for decoding the decision is presented below.

$$\delta_t(j) = \max_{\mathcal{Q}_t} P(q_1, \ldots, q_{t-1}, q_t = j | y_1, \ldots, y_t), \tag{9.1}$$

where $\mathcal{Q}_t$ indicates all possible decision sequences until time $t-1$. Then this can be recursively represented:

$$\begin{aligned} \delta_t(j) &= \max_{i=1:M} \delta_{t-1}(i) A_{ijk} \\ \Delta_t(j) &= \underset{i=1:M}{argmax}\, \delta_{t-1}(i) A_{ijk} \\ P^\star &= \max_{i=1:M} \delta_T(i) \end{aligned}$$

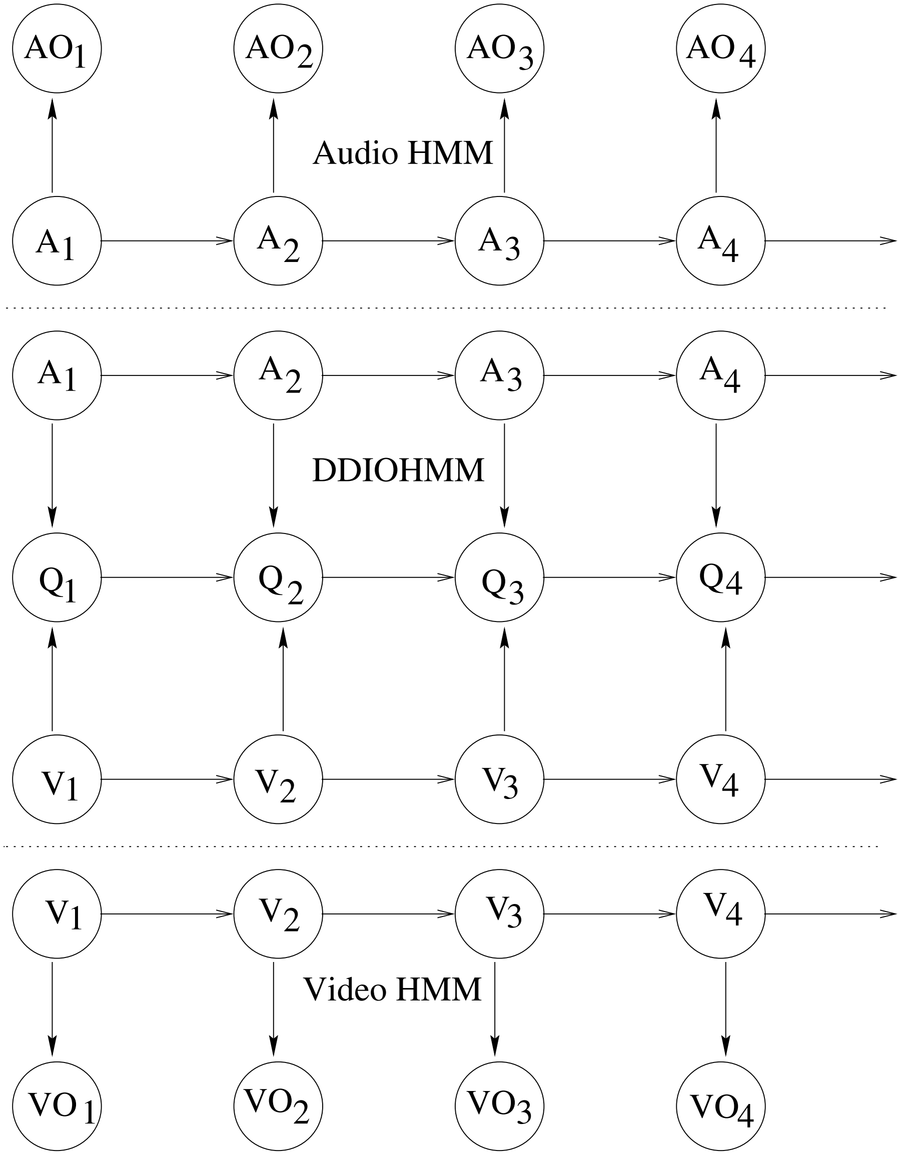


Figure 9.1. Hierarchical multimedia fusion. The media HMMs are responsible for mapping media observations to state sequences. The fusion model, which lies between the dotted lines, uses these state sequences as inputs and maps them to output decision sequences.

and $P^\star$ is the probability of the best sequence given the input and the parametric mapping A. We can then backtrack the best sequence $Q^\star$ as follows:

$$\begin{aligned} q_T^\star &= arg \max_{i=1:M} \delta_T(i) \\ &Backtracking \\ for \quad l &= 1 : T-1 \\ q_{T-l}^\star &= \Delta_{T-l+1}(q_{T-l+1}^\star). \end{aligned}$$

In the above algorithm, we have allowed the density of the duration of each decision state to be exponential. This may not be a valid assumption. To rectify this we now introduce the duration dependent input output Markov model (DDIOMM). Our approach in modeling duration in the DDIOMM is similar to that by Ramesh et al. [Ramesh and Wilpon, 1992].

This model, in its standard form, assumes that the probability of staying in a particular state decays exponentially with time. However, most audio-visual events have finite duration not conforming to the exponential density. Ramesh et al. [Ramesh and Wilpon, 1992] have shown that results can improve by specifically modeling the duration of staying in different states. In our current work, we show how one can enforce it in case of models like duration dependent IOMM.

Let us define a new mapping function:

$$A = \{A_{ijkd}\},\ i,j \in \{1,\ldots,M\},\ k \in \{1,\ldots,N\},\ d \in \{1,\ldots,D\}$$

where

$$A_{ijkd} = P(q_t = j | q_{t-1} = i, d_{t-1}(i) = d, y_t = k).$$

A can again be estimated from the training data by frequency counting. We now propose the algorithm to estimate the best decision sequence given A and the input sequence $\mathcal{Y}$. Let

$$\delta_t(j,d) = \max_{\mathcal{Q}_t} P(q_1,\ldots,q_{t-1},q_t = j, d_t(j) = d | y_1,\ldots,y_t) \tag{9.2}$$

where $\mathcal{Q}_t$ indicates all possible decision sequences until time $t-1$ and $d_t(i)$ indicates the duration in terms of discrete time samples for which the path has continued to be in the state q_t. This can be recursively computed as follows:

$$\begin{aligned} \delta_1(j,1) &= P(q_1|y_1) \\ \delta_1(j,d) &= 0 \ \ d > 1 \\ \delta_{t+1}(j,1) &= \max_{i=1:M\ i\neq j} \max_{d=1:D} \delta_t(i,d) A_{ijkd} \\ \delta_{t+1}(j,d+1) &= \delta_t(j,d) A_{jjkd}. \end{aligned}$$

Let

$$\Delta_t(j,i) = arg \max_{d=1:D} \delta_{t-1}(i,d)A_{ijkd} \quad 1 \leq i \leq j \quad i \neq j$$
$$\Psi_t(j) = arg \max_{i=1:M} \delta_{t-1}(i, \Delta_t(j,i))A_{ijk\Delta_t(j,i)}.$$

Finally,

$$\eta(i) = arg \max_{d=1:D} \delta_T(i,d) \quad 1 \leq i \leq M$$
$$P^\star = \max_{i=1:M} \delta_T(i, \eta(i)).$$

where $P^\star$ is the probability of the best sequence given the input and the parametric mapping A. We can then backtrack the best sequence $Q^\star$ as follows:

$$\begin{aligned} q_T^\star &= arg \max_{i=1:M} \delta_T(i,\eta(i)) \\ & Backtracking \\ x = \eta(q_T^\star) \quad t &= T \quad z = x \\ & while \quad t > 1 \\ q_{t-x+l}^\star = q_t^\star \quad l &= 1, \ldots, z-1 \\ q_{t-x}^\star &= \Psi_{t-x+1}(q_t^\star) \\ z &= \Delta_{t-x+1}(q_t^\star, q_{t-x}^\star) \\ t &= t - x \quad x = z. \end{aligned}$$

Using the above equations, one can decode the hidden state sequence. Each state (or the group of states) corresponds to the state of the environment or the particular semantic concept that is being modeled. In the next section, we compare the performance of these models with the traditional HMMs and show that one can get huge improvements.

3. Experimental Setup, Features, and Results

We compare the performance of our proposed algorithm with the IOMM as well as with the traditional HMM with their states being interpreted as decisions. We use the domain of movies and the audio-visual event *explosion* for comparison.

Data from a movie is digitized. We have over 10000 frames of video data and the corresponding audio data split in 9 clips. The data is labeled manually to construct the ground truth. Figure 9.2 show some typical frames of the video sequence.

From the visual stream we extract several features describing the color (HSV histograms, multiple order moments), structure (edge direction histogram), and texture (statistical measures of gray level co-occurrence matrices at multiple orientations) of the stream [Naphade and Huang, 2000a]. From the audio stream we extract 15 MFCC coefficients, 15 delta coefficients, and 2 energy coefficients [Naphade and Huang, 2000b]. As described in Section 9.2, we train HMMs for the positive as well as the negative hypothesis for the event *explosion*. HMMs for audio streams and video streams are separately trained. Each HMM has 3 states corresponding (intuitively) to the beginning, middle, and end state of the event. Using the pair of models for the positive and negative hypothesis we then segment each clip into two types of segments corresponding to the presence or absence of the event. Within each segment the best state sequence decoded by the Viterbi algorithm is available to us. This forms the input sequence $\mathcal{Y}$. Similarly, with h^s denoting the number of hypotheses for stream s, the total number of distinct symbols needed to describe the decision for each audio-visual frame jointly is given by $\prod_{s=1}^{S} h^s$. This forms the symbols of our decision sequence $\mathcal{Q}$.

Table 9.1. Comparing the overall classification error.

	HMM	IOMM	DDIOMM
Classification Error (%)	20.15	18.52	13.23

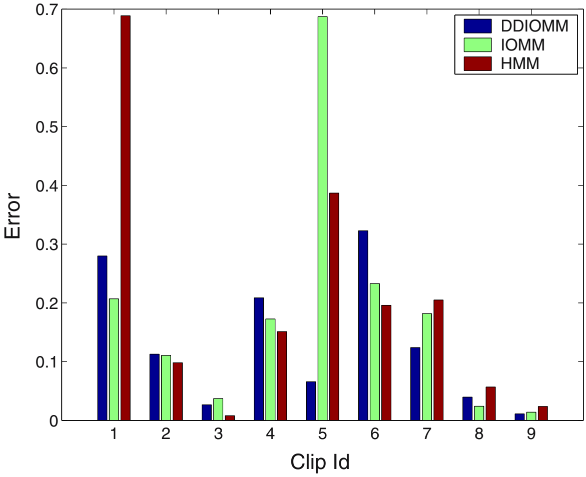
The results are reported using a leave-one-clip-out strategy. The quantum of time is a single video frame. To report performance objectively, we compare the prediction of the fusion algorithm for each video frame to our ground truth. Any difference between the two constitutes to a false alarm or mis-detection. We also compare the classification error of the three schemes. Figure 9.3(a) shows the error for each clip using the three schemes. Amongst the three schemes, the maximum error across all clips is minimum for the DDIOMM. Table 9.1 shows the overall classification error across all the clips. Clearly the overall classification error is the least for the DDIOMM.

We also compare the detection and false alarm rates of the three schemes. Figure 9.3(b) shows the detection and false alarm for the three schemes. Figure 9.3 and Table 9.1 thus show that the DDIOMM performs better event detection than the simple IOMM as well as the HMM.

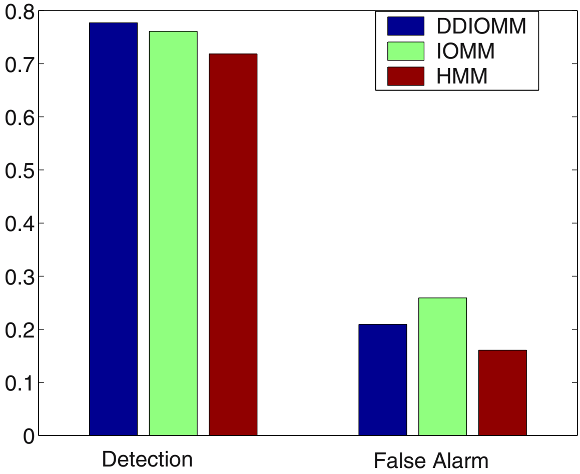
4. Summary

In this chapter, we have investigated a new model, the duration dependent input output Markov model (DDIOMM) for performing integration of intermediate decisions from different feature streams to detect events in Multimedia. The model provides a hierarchical mechanism to map media features to output decision sequences through intermediate state sequences. It also supports discrete non-exponential duration models for events. By combining these two fea-

Figure 9.2. Some typical frames from a Video Clip.



(a)



(b)

Figure 9.3. (a) Classification error for the nine video clips using the leave one clip out evaluation strategy. The maximum error for the DDIOMM is the least among the maximum error of the three schemes. (b) Comparing detection and false alarms. DDIOMM results in best detection performance.

tures in the framework of generative models for inference, we present a simple and efficient decision sequence decoding Viterbi algorithm. We demonstrate the strength of our model by experimenting with audio-visual data from movies and the audio-visual event *explosion*. Experiments comparing the DDIOMM with the IOMM as well as the HMM reveal that the DDIOMM results in lower classification error and improves detection.

As the results show, this models performs fairly well on the task of event detection in videos. This model and the LHMM presented in Chapter 8, fall in the category of what we call probabilistic classifiers. In the applications discussed and in many other applications in the past, it has been observed that probabilistic classifiers achieve very good performance in solving the real world problems. It is interesting to note, however, that these classifiers make assumptions which are often unrealistic or at least seems to be unrealistic. In general, one makes assumptions that audio is independent of video, or we assume that the data can be modeled using a HMM with only few hidden states modeling a one-Markov process (although we did relaxed this assumption in this chapter) and it is not clear if these assumptions will or does holds in practice.

The question that arises is how come these models still perform fairly well. Are we going to gain much if instead we decide to model the true distribution as against making these assumptions? In Chapter 2, we quantified these details and showed how the additional penalty on the classification error was related to the mismatch between the true and the modeled distribution.

Chapter 10

APPLICATION: FACIAL EXPRESSION RECOGNITION

The most expressive way humans display emotions is through facial expressions. Humans detect and interpret faces and facial expressions in a scene with little or no effort. Still, development of an automated system that accomplishes this task is rather difficult. There are several related problems: detection of an image segment as a face, extraction of the facial expression information, and classification of the expression (e.g., in emotion categories). A system that performs these operations accurately and in real time would be a major step forward in achieving a human-like interaction between the man and machine.

In this chapter, we compare the different approaches of the previous chapters for the design of a facial expression recognition system. Our experiments suggest that using the TAN classifiers and the stochastic structure search algorithm described in Chapter 7 outperform previous approaches using Bayesian network classifiers, or even compared to Neural networks. We also show experimentally that the learning the structure with the SSS algorithm holds the most promise when learning to classify facial expressions with labeled and unlabeled data.

1. Introduction

In recent years there has been a growing interest in improving all aspects of the interaction between humans and computers. This emerging field has been a research interest for scientists from several different scholastic tracks, i.e., computer science, engineering, psychology, and neuroscience. These studies focus not only on improving computer interfaces, but also on improving the actions the computer takes based on feedback from the user. Feedback from the user has traditionally been given through the keyboard and mouse. Other devices have also been developed for more application specific interfaces, such as joysticks, trackballs, datagloves, and touch screens. The rapid advance of technology in recent years has made computers cheaper and more powerful, and has made the use of microphones and PC-cameras affordable and easily available. The microphones and cameras enable the computer to “see” and “hear,” and to use this information to act. A good example of this is the “SmartKiosk” [Garg et al., 2000a].

It is argued that to truly achieve effective human-computer intelligent interaction (HCII), there is a need for the computer to be able to interact naturally with the user, similar to the way human-human interaction takes place.

Human beings possess and express emotions in everyday interactions with others. Emotions are often reflected on the face, in hand and body gestures, and in the voice, to express our feelings or likings. While a precise, generally agreed upon definition of emotion does not exist, it is undeniable that emotions are an integral part of our existence. Facial expressions and vocal emotions are commonly used in everyday human-to-human communication, as one smiles to show greeting, frowns when confused, or raises one’s voice when enraged. People do a great deal of inference from perceived facial expressions: “You *look* tired,” or “You *seem* happy.” The fact that we understand emotions and know how to react to other people’s expressions greatly enriches the interaction. There is a growing amount of evidence showing that emotional skills are part of what is called “intelligence” [Salovey and Mayer, 1990; Goleman, 1995]. Computers today, on the other hand, are still quite “emotionally challenged.” They neither recognize the user’s emotions nor possess emotions of their own.

Psychologists and engineers alike have tried to analyze facial expressions in an attempt to understand and categorize these expressions. This knowledge can be for example used to teach computers to recognize human emotions from video images acquired from built-in cameras. In some applications, it may not be necessary for computers to recognize emotions. For example, the computer inside an automatic teller machine or an airplane probably does not need to recognize emotions. However, in applications where computers take on a social role such as an “instructor,” “helper,” or even “companion,” it may enhance their functionality to be able to recognize users’ emotions. In her book, Picard [Picard, 1997] suggested several applications where it is beneficial for computers to recognize human emotions. For example, knowing the user’s emotions, the computer can become a more effective tutor. Synthetic speech with emotions in the voice would sound more pleasing than a monotonous voice. Computer “agents” could learn the user’s preferences through the users’ emotions. Another application is to help the human users monitor their stress level. In clinical settings, recognizing a person’s inability to express certain facial expressions may help diagnose early psychological disorders.

This chapter focuses on learning how to classify facial expressions with video as the input, using Bayesian networks. We have developed a real time facial expression recognition system [Cohen et al., 2002b; Sebe et al., 2002]. The system uses a model based non-rigid face tracking algorithm to extract motion features that serve as input to a Bayesian network classifier used for recognizing the different facial expressions.

There are two main motivations for using Bayesian network classifiers in this problem. The first is the ability to learn with unlabeled data and infer the class label even when some of the features are missing (e.g., due to failure in tracking because of occlusion). Being able to learn with unlabeled data is important for facial expression recognition because of the relatively small amount of available labeled data. Construction and labeling of a good database of images or videos of facial expressions requires expertise, time, and training of subjects and only a few such databases are available. However, collecting, without labeling, data of humans displaying expressions is not as difficult. The second motivation for using Bayesian networks is that it is possible to extend the system to fuse other modalities, such as audio, in a principled way by simply adding subnetworks representing the audio features.

2. Human Emotion Research

There is a vast body of literature on emotions. The multifaceted nature prevents a comprehensive review, we will review only what is essential in supporting this work. Recent discoveries suggest that emotions are intricately linked to other functions such as attention, perception, memory, decision making, and learning. This suggests that it may be beneficial for computers to recognize the human user's emotions and other related cognitive states and expressions. In this chapter, we concentrate on the expressive nature of emotion, especially those expressed in the voice and on the face.

2.1 Affective Human-computer Interaction

In many important HCI applications such as computer aided tutoring and learning, it is highly desirable (even mandatory) that the response of the computer takes into account the emotional or cognitive state of the human user. Emotions are displayed by visual, vocal, and other physiological means. Computers today can recognize much of what is said, and to some extent, who said it. But, they are almost completely in the dark when it comes to how things are said, the affective channel of information. This is true not only in speech, but also in visual communications despite the fact that facial expressions, posture, and gesture communicate some of the most critical information: how people feel. Affective communication explicitly considers how emotions can be recognized and expressed during human-computer interaction. Addressing the problem of affective communication, Bianchi-Berthouze and Lisetti [Bianchi-Berthouze and Lisetti, 2002] identified three key points to be considered when developing systems that capture affective information: embodiment (experiencing physical reality), dynamics (mapping experience and emotional state with its label), and adaptive interaction (conveying emotive response, responding to a recognized emotional state).

In most cases today, if you take a human-human interaction, and replace one of the humans with a computer, then the affective communication vanishes. Furthermore, this is not because people stop communicating affect - certainly we have all seen a person expressing anger at his machine. The problem arises because the computer has no ability to recognize if the human is pleased, annoyed, interested, or bored. Note that if a human ignored this information, and continued babbling long after we had yawned, we would not consider that person very intelligent. Recognition of emotion is a key component of intelligence [Picard, 1997]. Computers are presently affect-impaired. Furthermore, if you insert a computer (as a channel of communication) between two or more humans, then the affective bandwidth may be greatly reduced. Email may be the most frequently used means of electronic communication, but typically all of the emotional information is lost when our thoughts are converted to the digital media.

Research is therefore needed for new ways to communicate affect through computer-mediated environments. Computer-mediated communication today almost always has less affective bandwidth than “being there, face-to-face”. The advent of affective wearable computers, which could help amplify affective information as perceived from a person’s physiological state, are but one possibility for changing the nature of communication.

2.2 Theories of Emotion

There is little agreement about a definition of emotion. Many theories of emotion have been proposed. Some of these could not be verified until recently when measurement of some physiological signals become available. In general, emotions are short-term, whereas moods are long-term, and temperaments or personalities are very long-term [Jenkins et al., 1998]. A particular mood may sustain for several days, and temperament for months or years. Finally, emotional disorders can be so disabling that people affected are no longer able to lead normal lives.

Darwin [Darwin, 1890] held an ethological view of emotional expressions, arguing that the expressions from infancy and lower life forms exist in adult humans. Following the *Origin of Species* he wrote *The Expression of the Emotions in Man and Animals*. According to him, emotional expressions are closely related to survival. Thus in human interactions, these nonverbal expressions are as important as the verbal interaction.

James [James, 1890] viewed emotions not as *causes* but as *effects*. Situations arise around us which cause changes in physiological signals. According to James, "the bodily changes follow directly the perception of the exciting fact, and that our feeling of the same changes as they occur *is* the emotion." Carl Lange proposed a similar theory independently at around the same time. This is often referred to as the "James-Lange" theory of emotion. Cannon [Cannon, 1927], contrary to James, believed that emotions are first felt, then exhibited outwardly causing certain behaviors.

Despite the many theories, it is evident that people display these expressions to various degrees. One frequently studied task is the judgment of emotions—how well can human observers tell the emotional expressions of others, in the voice, on the face, etc? Related questions are: Do these represent their true emotions? Can they be convincingly portrayed? How well can people conceal their emotions? In such tasks, researchers often use two different methods to describe the emotions.

One approach is to label the emotions in discrete categories, i.e., human judges must choose from a prescribed list of word labels, such as *joy, fear, love, surprise, sadness*, etc. One problem with this approach is that the stimuli may contain blended emotions. Also, the choice of words may be too restrictive, or culturally dependent.

Another way is to have multiple dimensions or scales to describe emotions. Instead of choosing discrete labels, observers can indicate their impression of each stimulus on several continuous scales, for example, pleasant–unpleasant, attention–rejection, simple–complicated, etc. Two common scales are valence and arousal. Valence describes the pleasantness of the stimuli, with positive (or pleasant) on one end, and negative (or unpleasant) on the other. For example, *happiness* has a positive valence, while *disgust* has a negative valence. The other dimension is arousal or activation. For example, *sadness* has low arousal, whereas *surprise* has high arousal level. The different emotional labels could be plotted at various positions on a two-dimensional plane spanned by these two axes to construct a 2D emotion model [Lang, 1995]. Scholsberg [Schlosberg, 1954] suggested a three-dimensional model in which he had *attention–rejection* in addition to the above two.

Another interesting topic is how the researchers managed to obtain data for observation. Some people used posers, including professional actors and non-actors. Others attempted to induce emotional reactions by some clever means. For example, Ekman showed stress-inducing film of nasal surgery in order to get the disgusted look on the viewers' faces. Some experimenter even dumped water on the subjects or fired blank shots to induce surprise, while others used clumsy technicians who made rude remarks to arouse fear and anger [Hilgard et al., 1971]. Obviously, some of these are not practical ways of acquiring data. After studying acted and natural expressions, Ekman concluded that expressions can be convincingly portrayed [Ekman, 1982].

A legitimate question that should be considered when doing multimodal emotion recognition is how much information does the face, as compared to voice, speech, and body movement, provide about emotion. Most experimenters found that the face is more accurately judged, produces higher agreement, or correlates better with judgments based on full audiovisual input than on voice or speech input [Mehrabian, 1968; Ekman, 1982]. Ekman [Ekman, 1982] found that the relative weight given to facial expression, speech, and body cues depend both on the judgment task (e.g., rating the stimulus subject's dominance, sociability, or relaxation) and the conditions in which the behavior occurred (e.g., subjects frankly describing positive reactions to a pleasant film or trying to conceal negative feelings aroused by a stressful film).

The whole question of how much information is conveyed by "separate" channels may inevitably be misleading. There is no evidence that individuals in actual social interaction selectively attend to another person's face, body, voice, or speech or that the information conveyed by these channels is simply additive. The central mechanisms directing behavior cut across the channels, so that, for example, certain aspects of face, body, voice, and speech are more spontaneous and others are more closely monitored and controlled. It might well be that observers selectively attend not to a particular channel but to a particular type of information (e.g., cues to emotion, deception, or cognitive activity), which may be available within several channels. No investigator has yet explored this possibility or the possibility that different individuals may typically attend to different types of information.

2.3 Facial Expression Recognition Studies

The mounting evidence of the importance of emotions in human-human interaction provided the basis for researchers in the engineering and computer science communities to develop automatic ways for computers to recognize emotional expression, as a goal towards achieving human-computer intelligent interaction. The labeling of emotions into different states led most research to use pattern recognition approaches for recognizing emotions, using different modalities as inputs to the emotion recognition models. Next we review some of these works.

Since the early 1970s, Paul Ekman and his colleagues have performed extensive studies of human facial expressions [Ekman, 1994]. They found evidence to support universality in facial expressions. These "universal facial expressions" are those representing happiness, sadness, anger, fear, surprise, and disgust. They studied facial expressions in different cultures, including preliterate cultures, and found much commonality in the expression and recognition of emotions on the face. However, they observed differences in expressions as well, and proposed that facial expressions are governed by "display rules" in different social contexts. For example, Japanese subjects and American subjects showed similar facial expressions while viewing the same stimulus film. However, in the presence of authorities, the Japanese viewers were more reluctant to show their real expressions. Matsumoto [Matsumoto, 1998] reported the discovery of a seventh universal facial expression: contempt. Babies seem to exhibit a wide range of facial expressions without being taught, thus suggesting that these expressions are innate [Izard, 1994].

Ekman and Friesen [Ekman and Friesen, 1978] developed the Facial Action Coding System (FACS) to code facial expressions where movements on the face are described by a set of action units (AUs). Each AU has some related muscular basis. Each facial expression may be described by a combination of AUs. Figure 10.1 shows some of the key facial muscles on the face [Faigin, 1990]. The muscle movements (contractions) produce facial expressions. For example, the *corrugator* is also known as the "frowning muscle," *zygomatic major* is responsible for smiling, and *lavator labii superioris* produces "sneering." Table 10.1 lists some example action units. Each facial expression may be described by a combination of AUs. This system of coding facial expressions is done manually by following a set prescribed rules. The inputs are still images of facial expressions, often at the peak of the expression. This process is very time-consuming.

Table 10.1. Some example action units [Ekman and Friesen, 1978].

AU number	FACS name	Muscular basis
1	Inner brow raiser	Frontalis, pars medialis
2	Outer brow raiser	Frontalis, pars lateralis
5	Upper lid raiser	Levator palpebrae superioris
11	Nasolabial furrow	Zygomatic minor
12	Lip corner puller	Zygomatic major
20	Lip stretcher	Risorious

Ekman's work inspired many researchers to analyze facial expressions by means of image and video processing. By tracking facial features and measuring the amount of facial movement, they attempt to categorize different facial expressions. Recent work on facial expression analysis and recognition [Mase, 1991; Ueki et al., 1994; Lanitis et al., 1995b; Black and Yacoob, 1995; Rosenblum et al., 1996; Essa and Pentland, 1997; Otsuka and Ohya, 1997a; Lien, 1998; Nefian and Hayes, 1999b; Martinez, 1999; Oliver et al., 2000; Cohen et al., 2003b; Chen, 2000; Cohen et al., 2003a] has used these "basic expressions" or a subset of them. The recent surveys in the area [Fasel and Luettin, 2003; Pantic and Rothkrantz, 2000; Pantic and Rothkrantz, 2003] provide an in-depth review of many of the research done in automatic facial expression recognition in recent years.

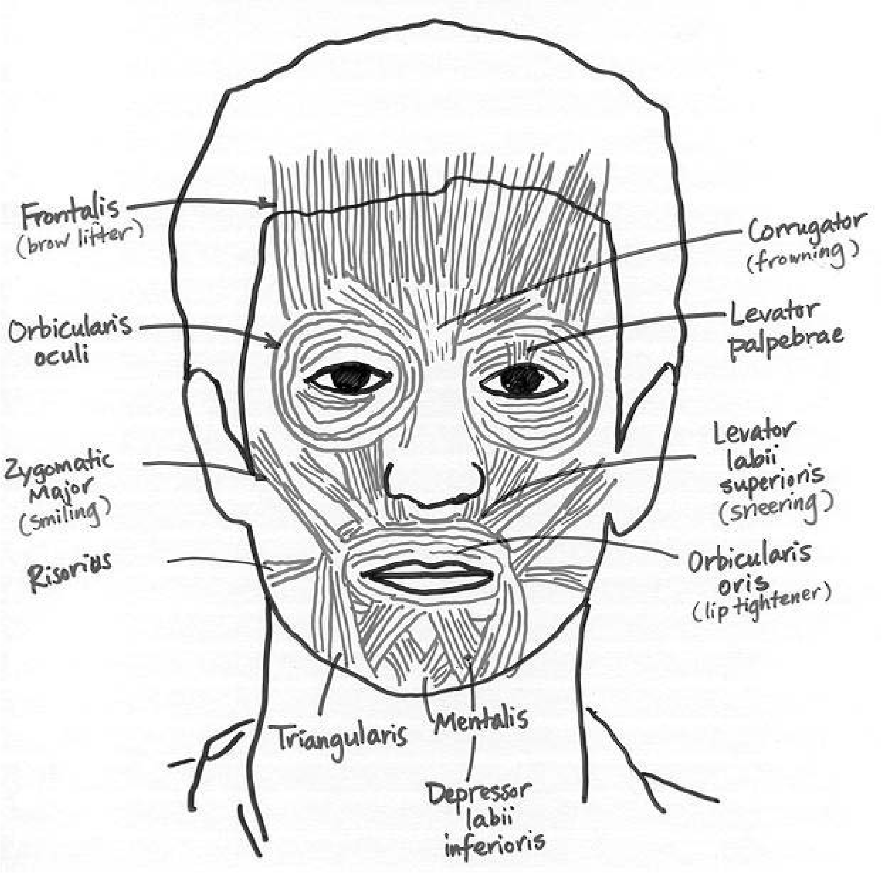


Figure 10.1. Some key facial muscles (adapted from [Faigin, 1990]).

Recent work in computer-assisted quantification of facial expressions did not start until the 1990s. Mase [Mase, 1991] used optical flow (OF) to recognize facial expressions. He was one of the first to use image processing techniques to recognize facial expressions. Lanitis et al. [Lanitis et al., 1995b] used a flexible shape and appearance model for image coding, person identification, pose recovery, gender recognition and facial expression recognition. Black and Yacoob [Black and Yacoob, 1995] used local parameterized models of image motion to recover non-rigid motion. Once recovered, these parameters are fed to a rule-based classifier to recognize the six basic facial expressions. Yacoob and Davis [Yacoob and Davis, 1996] computed optical flow and used similar rules to classify the six facial expressions. Rosenblum et al. [Rosenblum et al., 1996] also computed optical flow of regions on the face, then applied a radial basis function network to classify expressions. Essa and Pentland [Essa and Pentland, 1997] also used an optical flow region-based method to recognize expressions. Otsuka and Ohya [Otsuka and Ohya, 1997a] first computed optical flow, then computed their 2D Fourier transform coefficients, which were then used as feature vectors for a hidden Markov model (HMM) to classify expressions. The trained system was able to recognize one of the six expressions near realtime (about 10 Hz). Furthermore, they used the tracked motions to control the facial expression of an animated Kabuki system [Otsuka and Ohya, 1997b]. A similar approach, using different features was used by Lien [Lien, 1998]. Nefian and Hayes [Nefian and Hayes, 1999b] proposed an embedded HMM approach for face recognition that uses an efficient set of observation vectors based on the DCT coefficients. Martinez [Martinez, 1999] introduced an indexing approach based on the identification of frontal face images under different illumination conditions, facial expressions, and occlusions. A Bayesian approach was used to find the best match between the local observations and the learned local features model and an HMM was employed to achieve good recognition even when the new conditions did not correspond to the conditions previously encountered during the learning phase. Oliver et al. [Oliver et al., 2000] used lower face tracking to extract mouth shape features and used them as inputs to an HMM based facial expression recognition system (recognizing neutral, happy, sad, and an open mouth). Chen [Chen, 2000] used a suite of static classifiers to recognize facial expressions, reporting on both person-dependent and person-independent results. Cohen et al. [Cohen et al., 2003b] describe classification schemes for facial expression recognition in two types of settings: dynamic and static classification. The static classifiers classify a frame in a video to one of the facial expression categories based on the tracking results of that frame. In this setting, the authors learn the structure of Bayesian networks classifiers using as input 12 motion units given by a face tracking system. The authors also use schemes that utilize data that are unlabeled and cheap to obtain, in conjunction with (expensively) labeled data [Cohen et al., 2003a; Cohen et al., 2004]. For the dynamic setting, they used a multi-level HMM classifier that combines the temporal information and allows not only to perform the classification of a video segment to the corresponding facial expression, as in the previous works on HMM based classifiers, but also to automatically segment an arbitrary long sequence to the different expression segments without resorting to heuristic methods of segmentation.

These methods are similar in the general sense that they first extract some features from the images, then these features are fed into a classification system, and the outcome is one of the preselected emotion categories. They differ mainly in the features extracted from the video images or the processing of video images to classify emotions. The video processing falls into two broad categories. The first is “feature-based,” where one tries to detect and track specific features such as the corners of the mouth, eyebrows, etc.; the other approach is “region-based” in which facial motions are measured in certain regions on the face such as the eye/eyebrow and mouth regions. People have used different classification algorithms to categorize these emotions. In Table 10.2, we compare several facial expression recognition algorithms. In general, these algorithms perform well compared to trained human recognition of about 87% as reported by Bassili [Bassili, 1979].

Table 10.2. Comparisons of facial expression recognition algorithms.

Author	Processing	Classification	Number of Categories	Number of Subjects	Performance
Mase	optical flow	kNN	4	1	86%
Black & Yacoob	parametric model	rule-based	6	40	92%
Yacoob & Davis	optical flow	rule-based	6	32	95%
Rosenblum et al.	optical flow	neural networks	2	32	88%
Essa & Pentland	optical flow	distance-based	5	8	98%
Otsuka & Ohya	2D FT of optical flow	HMM	6	4	93%
Lanitis et al.	appearance model	distance-based	7	-	74%
Chen	appearance model	Winnow	6	5	86%
Cohen et al.	appearance model	Bayesian networks	7	5+53	83%

In contrast to the classification methods described above, Ueki et al. [Ueki et al., 1994] extracted AUs and used neural networks (NN) to analyze the emotions, mapping seventeen AUs to two dimensions using an identity mapping network, and this showed resemblance of the 2D psychological emotion models. Later on, Morishima [Morishima, 1995] proposed a 3D emotion model in order to deal with transitions between emotions, and claimed correlation to the 3D psychological emotion model [Schlosberg, 1954].

Another interesting thing to point out is the problem of the commonly confused categories in the six basic expressions. As reported by Ekman, *anger* and *disgust* are commonly confused in judgment studies. Also, *fear* and *surprise* are commonly confused. The reason why these confusions occur is because they share many similar facial actions[Ekman and Friesen, 1978]. *Surprise* is sometimes mistaken for *interest*, but not the other way around. In the computer recognition studies, some of these confusions are observed [Black and Yacoob, 1995; Yacoob and Davis, 1996; Cohen et al., 2003b].

3. Facial Expression Recognition System

Our real time facial expression recognition system is composed of face tracking algorithm which outputs a vector of motion features of certain regions of the face. The features are used as inputs to a Bayesian network classifier. We describe these components in the following section. A snap shot of the system, with the face tracking and recognition result is shown in Figure 10.2.

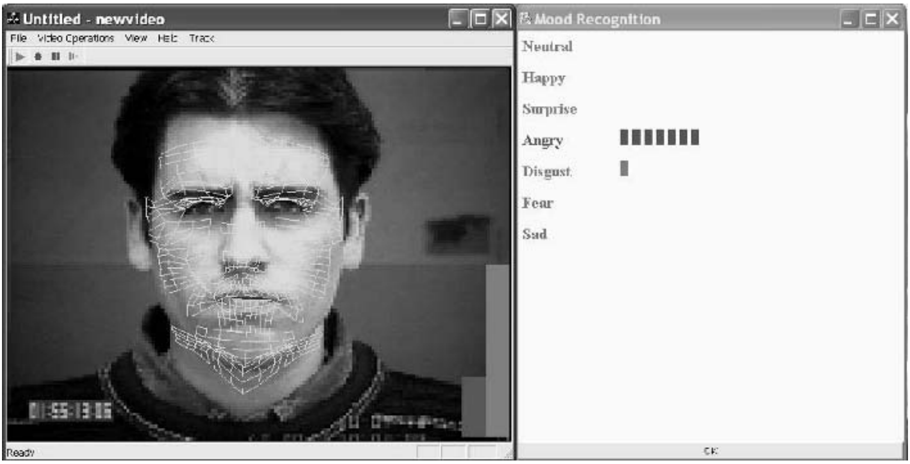


Figure 10.2. A snap shot of our realtime facial expression recognition system. On the right side is a wireframe model overlayed on a face being tracked. On the left side the correct expression, Angry, is detected (the bars show the relative probability of Angry compared to the other expressions). The subject shown is from the Cohn-Kanade database [Kanade et al., 2000].

3.1 Face Tracking and Feature Extraction

The face tracking we use in our system is based on a system developed by Tao and Huang [Tao and Huang, 1998] called the piecewise Bezier volume deformation (PBVD) tracker.

The face tracker uses a model-based approach where an explicit 3D wireframe model of the face is constructed. In the first frame of the image sequence, landmark facial features such as the eye corners and mouth corners are selected interactively. The generic face model is then warped to fit the selected facial features. The face model consists of 16 surface patches embedded in Bezier volumes. The surface patches defined this way are guaranteed to be continuous and smooth. The shape of the mesh can be changed by changing the locations of the control points in the Bezier volume. Before describing the Bezier volume, we begin with the Bezier curve.

Given a set of $n+1$ control points $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n$, the corresponding Bezier curve (or Bernstein-Bezier curve) is given by

$$\mathbf{x}(u) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(u) = \sum_{i=0}^{n} \mathbf{b}_i \binom{n}{i} u^i (1-u)^{n-i} \tag{10.1}$$

where the shape of the curve is controlled by the control points $\mathbf{b}_i$ and $u \in [0, 1]$. As the control points are moved, a new shape is obtained according to the Bernstein polynomials $B_i^n(u)$ in Equation (10.1). The displacement of a point on the curve can be described in terms of linear combinations of displacements of the control points.

The Bezier volume is a straight-forward extension of the Bezier curve and is defined by the next equation written in matrix form

$$\mathbf{V} = \mathbf{BD}, \tag{10.2}$$

where $\mathbf{V}$ is the displacement of the mesh nodes, $\mathbf{D}$ is a matrix whose columns are the control point displacement vectors of the Bezier volume, and $\mathbf{B}$ is the mapping in terms of Bernstein polynomials. In other words, the change in the shape of the face model can be described in terms of the deformations in $\mathbf{D}$.

Once the model is constructed and fitted, head motion and local deformations of the facial features such as the eyebrows, eyelids, and mouth can be tracked. First the 2D image motions are measured using template matching between frames at different resolutions. Image templates from the previous frame and from the very first frame are both used for more robust tracking. The measured 2D image motions are modeled as projections of the true 3D motions onto the image plane. From the 2D motions of many points on the mesh, the 3D motion can be estimated by solving an overdetermined system of equations of the projective motions in the least squared sense. Figure 10.3 shows four frames of tracking result with the meshes overlaid on the face.

The recovered motions are represented in terms of magnitudes of some predefined motion of various facial features. Each feature motion corresponds to a simple deformation on the face, defined in terms of the Bezier volume control parameters. We refer to these motions vectors as motion-units (MU's). Note that they are similar but not equivalent to Ekman's AU's, and are numeric in nature, representing not only the activation of a facial region, but also the direction and intensity of the motion. The MU's used in the face tracker are shown in Figure 10.4 and are described in Table 10.3.

Each facial expression is modeled as a linear combination of the MU's:

$$\mathbf{V} = \mathbf{B}\left[\mathbf{D}_0 \mathbf{D}_1 \ldots \mathbf{D}_m\right] \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_m \end{bmatrix} = \mathbf{BDP} \tag{10.3}$$

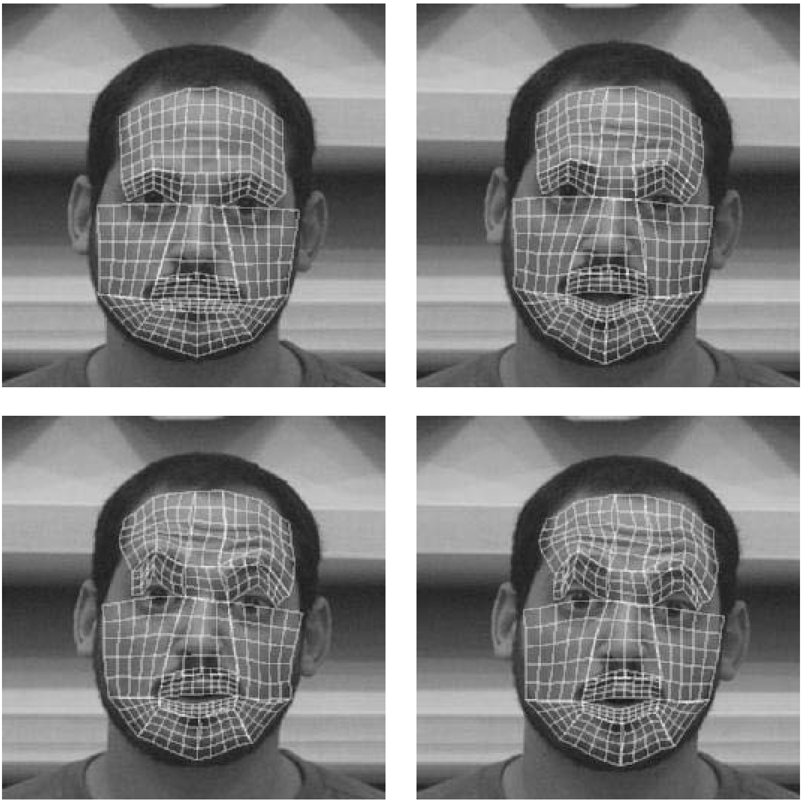


Figure 10.3. The wireframe model overlaid on a face being tracked.

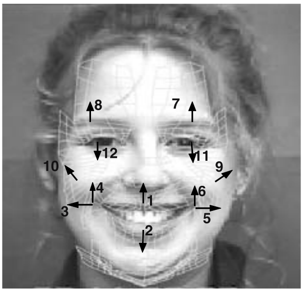


Figure 10.4. The facial motion measurements.

Table 10.3. Motion units used in our face tracker.

AU	Description
1	vertical movement of the center of upper lip
2	vertical movement of the center of lower lip
3	horizontal movement of left mouth corner
4	vertical movement of left mouth corner
5	horizontal movement of right mouth corner
6	vertical movement of right mouth corner
7	vertical movement of right brow
8	vertical movement of left brow
9	lifting of right cheek
10	lifting of left cheek
11	blinking of right eye
12	blinking of left eye

where each of the $\mathbf{D}_i$ corresponds to an MU, and the p_i are the corresponding magnitudes (or coefficients) of each deformation. The overall motion of the head and face is

$$\mathbf{R}(\mathbf{V}_0 + \mathbf{BDP}) + \mathbf{T} \tag{10.4}$$

where $\mathbf{R}$ is the 3D rotation matrix, $\mathbf{T}$ is the 3D translation matrix, and $\mathbf{V}_0$ is the initial face model.

The MU's are used as the basic features for the classification scheme described in the next sections.

3.2 Bayesian Network Classifiers: Learning the "Structure" of the Facial Features

The use of Bayesian networks as the classifier for recognizing facial expressions has been first suggested by Chen et al. [Chen, 2000], who used Naive Bayes classifiers and to recognize the facial expressions from the same MUs. In [Sebe et al., 2002], we proposed changing the assumption on the distribution of the features from Gaussian to Cauchy so as to enhance the performance of the Naive Bayes classifier.

When modeling the describe facial motion features, it is very probable that the conditional independence assumption of the Naive Bayes classifier is incorrect. As such, learning the dependencies among the facial motion units could potentially improve classification performance, and could provide insights as to the "structure" of the face, in terms of strong or weak dependencies between the different regions of the face, when subjects display facial expressions. With unlabeled data, the analysis of the previous chapters indicates that learning the structure is even more critical compared to the supervised case. As such, we employ the different methods that have been suggested in Chapter 7, in particular we are interested in using the TAN and SSS algorithms, observing performance for both the supervised and the semi-supervised cases.

4. Experimental Analysis

In the following experiments we compare the different approaches discussed in Chapter 7 for facial expression recognition. We initially consider experiments where all the data is labeled and show limited experiments for person dependent tests and then show experiments for the more general problem of person independent expression recognition. In the second part of the test, we investigate the effect of using both labeled and unlabeled data.

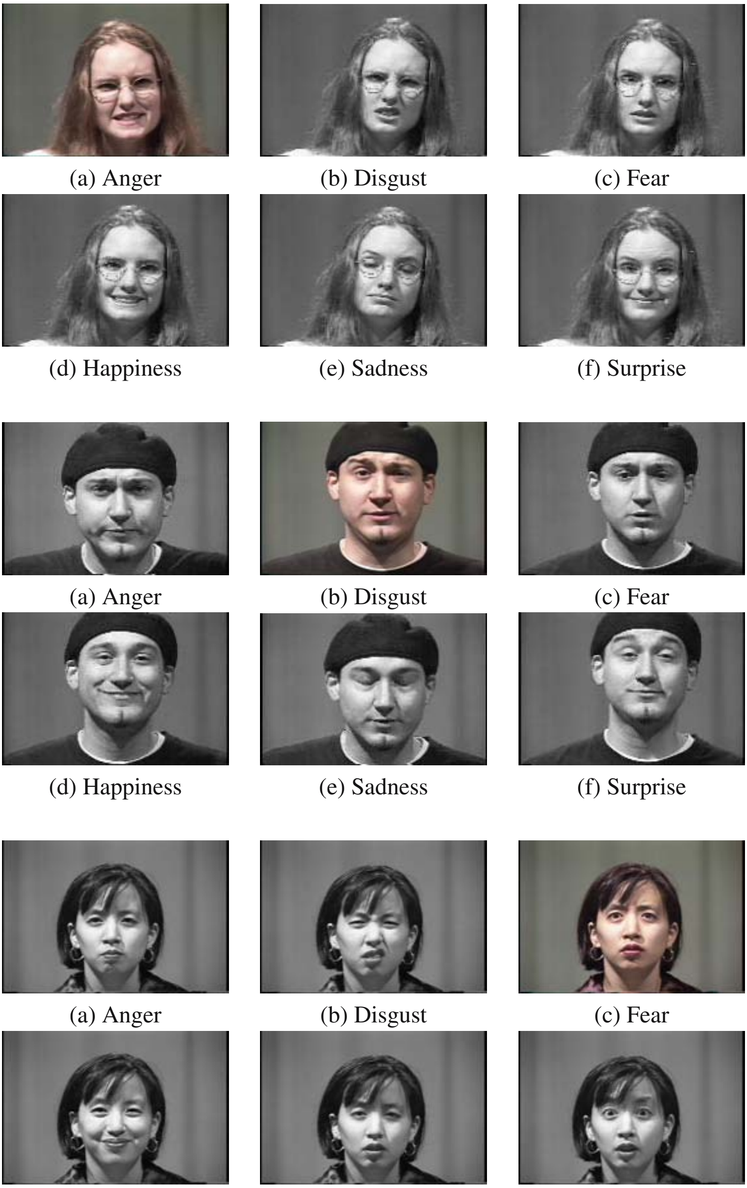
We use two different databases, a database collected by Chen and Huang [Chen, 2000] and the Cohn-Kanade AU coded facial expression database [Kanade et al., 2000].

The first is a database of subjects that were instructed to display facial expressions corresponding to the six types of emotions. All the tests of the algorithms are performed on a set of five people, each one displaying six sequences of each one of the six emotions, starting and ending at the neutral expression. The data collection method is described in detail in [Chen, 2000]. All the tests of the algorithms are performed on a set of five people, each one displaying six sequences of each one of the six emotions, and always coming back to a neutral state between each emotion sequence.

Each video sequence was used as the input to the face tracking algorithm described in Section 10.3.1. The video sampling rate was 30 Hz, and a typical emotion sequence is about 70 samples long (∼2s). Figure 10.5 shows one frame of each subject.

The data was collected in an open recording scenario, where the person was asked to display the expression corresponding to the emotion being induced. This is of course not the ideal way of collecting emotion data. The ideal way would be using a hidden recording, inducing the emotion through events in the normal environment of the subject, not in a studio [Sebe et al., 2004]. The main problem with collecting the data this way is the impracticality of it and the ethical issue of hidden recording.

The Cohn-Kanade database [Kanade et al., 2000] consists of expression sequences of subjects, starting from a Neutral expression and ending in the peak of the facial expression. There are 104 subjects in the database. Because not all of the six facial expressions sequences were available to us for some of the subjects, we used a subset of 53 subjects, for which at least four of the sequences were available. For each subject there is at most one sequence per expression with an average of 8 frames for each expression. Figure 10.6 shows some examples of subjects from the database. A summary of both databases is presented in Table 10.4.



(a) Anger (b) Disgust (c) Fear

(d) Happiness (e) Sadness (f) Surprise

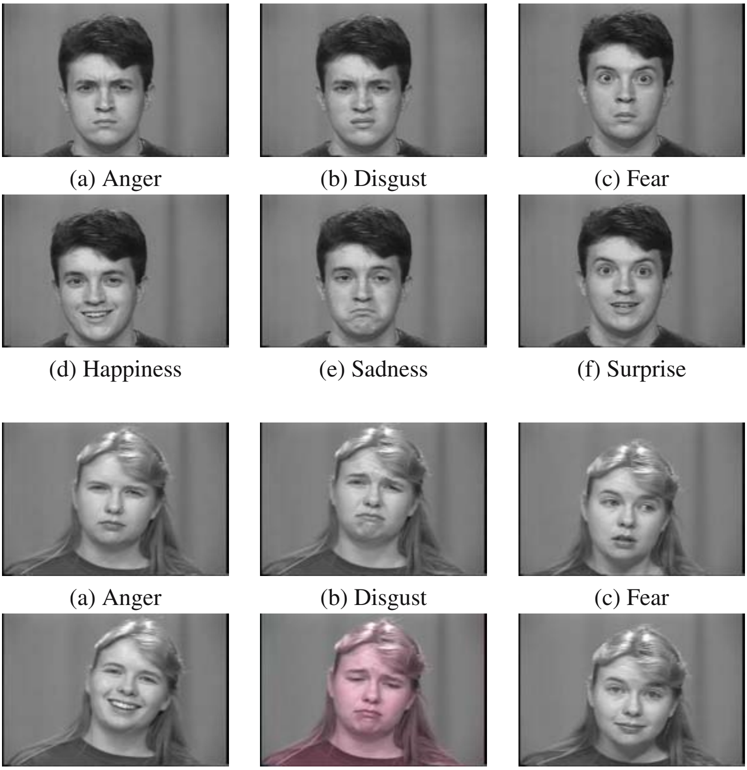
(a) Anger (b) Disgust (c) Fear

(d) Happiness (e) Sadness (f) Surprise

(a) Anger (b) Disgust (c) Fear

(d) Happiness (e) Sadness (f) Surprise

Figure 10.5. Examples of images from the video sequences.



(a) Anger (b) Disgust (c) Fear

(d) Happiness (e) Sadness (f) Surprise

(a) Anger (b) Disgust (c) Fear

(d) Happiness (e) Sadness (f) Surprise

Figure 10.5 (continued). Examples of images from the video sequences.

Table 10.4. Summary of the databases.

Database	Subjects	Sequences per expression	Sequences per subject per expression	average frames per expression
Our DB	5	30	6	70
Cohn-Kanade DB	53	53	1	8

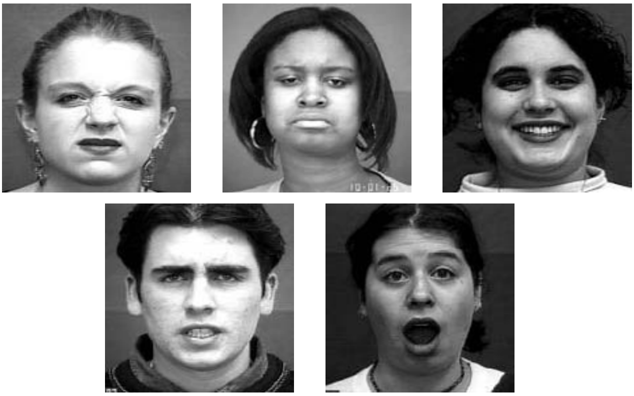


Figure 10.6. Examples of images from the from the Cohn-Kanade database used in the experiments (printed with permission from the researchers).

We measure the accuracy with respect to the classification result of each frame, where each frame in the video sequence was manually labeled to one of the expressions (including neutral). This manual labeling can introduce some 'noise' in our classification because the boundary between Neutral and the expression of a sequence is not necessarily optimal, and frames near this boundary might cause confusion between the expression and the Neutral. A different labeling scheme is to label only some of the frames that are around the peak of the expression leaving many frames in between unlabeled. We did not take this approach because a real-time classification system would not have this information available to it.

4.1 Experimental Results with Labeled Data

We start with experiments using all data as labeled. This can be viewed as an upper bound on the performance of the classifiers trained with most of the labels removed. We first perform person dependent tests, also comparing different assumptions other than different structures. Then we perform person independent tests, with both databases, showing the best performance is attained with the SSS algorithm. While our focus is on the performance of Bayesian network classifiers for this problem, we also compare the results with that of a artificial Neural network (ANN) based classifiers [Cohen et al., 2003c], so as to show that the Bayesian network classifiers are not inferior to other types of classifiers.

4.1.1 Person-dependent Tests

A person-dependent test is first tried. We test for person dependent results only using the Chen-Huang database since the Cohn-Kanade database has only one sequence per expression per subject, making it impossible to do person dependent tests. We train classifiers for each subject, leaving out some of the subject's data for testing. Tables 10.5 show the recognition rate of each subject and the (weighted) average recognition rate of the classifiers for all five subjects.

Table 10.5. Person-dependent facial expression recognition accuracies (in %).

Subject	NB-Gaussian	NB-Cauchy	NB-discrete	TAN	SSS	ANN
1	80.97	81.69	89.56	92.48	92.95	82.37
2	87.09	84.54	87.77	91.22	90.31	85.23
3	69.06	71.74	85.10	89.62	90.74	81.17
4	82.50	83.05	87.03	91.49	91.77	80.05
5	77.19	79.25	77.87	89.36	87.25	71.78
Average	79.21	79.31	85.58	90.92	90.48	80.81

Table 10.6. Person-dependent confusion matrix using the TAN based classifier.

Emotion	Neutral	Happy	Surprise	Anger	Disgust	Fear	Sad
Neutral	93.05	0.61	0.90	1.34	0.84	1.87	1.37
Happy	3.40	93.87	0.41	0.21	0.91	0.33	0.87
Surprise	12.85	0.48	82.64	1.70	0.0	0.63	1.70
Anger	7.98	0.22	0.0	88.44	1.40	1.25	0.71
Disgust	7.51	0.63	0.91	1.05	87.50	1.75	0.65
Fear	14.38	0.36	1.0	1.02	0.0	83.24	0.0
Sad	12.61	0.0	0.31	0.57	0.61	0.0	85.90

We compare the results of five assumptions, both on structure and the distributions of each feature. We first see that under the Naive-Bayes assumption, using entropic discretization (NB-discrete) outperforms both the Gaussian and Cauchy Naive Bayes classifiers. Compared to the three Naive Bayes classifiers, learning dependencies, either using TAN or using SSS, significantly improves the classification performance. We do not see any significant differences between SSS and TAN classifiers because the SSS algorithm could not search for structures much more complicated than the TAN, due to the limited size training set (higher complexity classifiers would overfit the training data). We

Table 10.7. Person-dependent confusion matrix using the SSS based classifier.

Emotion	Neutral	Happy	Surprise	Anger	Disgust	Fear	Sad
Neutral	92.27	0.85	0.83	1.38	0.78	2.07	1.82
Happy	5.64	90.76	0.41	0.25	1.09	1.02	0.81
Surprise	14.13	0.48	83.69	1.22	0.0	0.0	0.48
Anger	7.84	0.0	0.0	88.75	1.68	1.00	0.72
Disgust	4.72	0.29	0.62	1.43	91.95	0.81	0.18
Fear	10.04	0.0	1.36	1.00	0.0	87.27	0.33
Sad	13.10	0.0	0.0	0.52	1.43	0.0	84.95

also see that ANN outperform the continuous Naive Bayes classifiers, but are significantly inferior to TAN and SSS based classifiers.

The confusion matrices for the the TAN and the SSS based classifiers are presented in Table 10.6 and Table 10.7. The analysis of the confusion between different emotions shows that for both classifiers, most of the confusion of the classes is with the Neutral class. This can be attributed to the arbitrary labeling of each frame in the expression sequence. The first and last few frames of each sequence are very close to the Neutral expression and thus are more prone to become confused with it. We also see that most expression do not confuse with Happy, but more often get confused with each other.

4.1.2 Person-independent Tests

We perform person independent tests by partitioning the data such that the sequences of some subjects are used as the test sequences and the sequences of the remaining subjects are used as training sequences. Table 10.8 shows the recognition rate of the test for all classifiers, using only the discretized features. The classifier learned with the SSS algorithm outperforms both the NB and TAN classifiers, while ANN do not perform well compared to all the others.

Table 10.8. Recognition rate (%) for person-independent test.

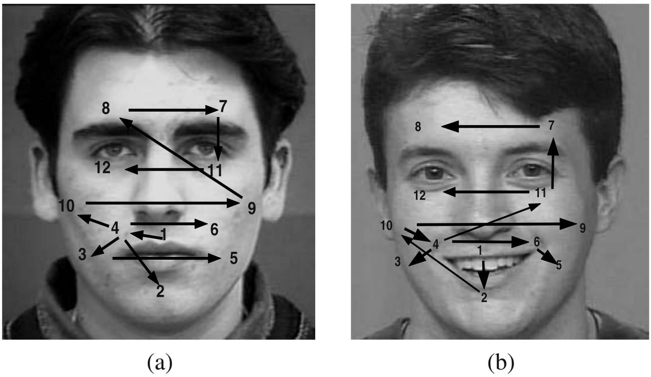
	NB	TAN	SSS	ANN
Chen-Huang Database	71.78	80.31	83.62	66.44
Cohn-Kandade Database	77.70	80.40	81.80	73.81

Table 10.9 shows the confusion matrices for SSS classifier trained with the Cohn-Kanade database. We see that Happy, Surprise and Neutral are detected with high accuracy, and other expressions are confused mostly with Neutral. Here the differences in the intensity of the expressions among the different subjects played a role in the confusion among the different expressions.

Table 10.9. Person-independent average confusion matrix using the SSS classifier (Cohn-Kanade Database).

Emotion	Neutral	Happy	Surprise	Anger	Disgust	Fear	Sad
Neutral	92.39	1.84	0.79	2.10	0.00	0.52	2.36
Happy	4.14	84.14	1.38	1.38	2.07	6.21	0.69
Surprise	3.19	0.00	91.49	0.00	0.00	1.06	4.26
Anger	13.46	1.92	0.96	71.15	7.69	0.96	3.85
Disgust	22.81	1.75	0.00	8.77	64.91	1.75	0.00
Fear	18.18	10.10	7.07	0.00	1.01	58.59	5.05
Sad	13.33	1.67	5.00	5.00	0.00	0.83	74.17

It is also informative to look at the structures that were learned from data. Figure 10.7 shows two learned tree structure of the features (our Motion Units) one learned using the Cohn-Kanade database and the second from the Chen-Huang database. The arrows are from parents to children MUs. In both tree structures we see that the algorithm produced structures in which the bottom half of the face is almost disjoint from the top portion, except for a link between MU9 and MU8 in the first and a weak link between MU4 and MU11 in the second.



(a) (b)

Figure 10.7. Two learned TAN structures for the facial features, (a) using the Cohn-Kanade database, (b) using the Chen-Huang database.

4.2 Experiments with Labeled and Unlabeled Data

We perform person-independent experiments with labeled and unlabeled data. We first partition the data to a training set and test set (2/3 training 1/3 for testing), choose by random a portion of the training set and remove the labels (see Table 10.10). This procedure ensures that the distribution of the labeled and the unlabeled sets are the same.

Table 10.10. Datasets used for facial expression recognition with labeled and unlabeled data.

Dataset	Train		Test
	# labeled	# unlabeled	
Cohn-Kanade	200	2980	1000
Chen-Huang	300	11982	3555

We then train Naive Bayes and TAN classifiers, using just the labeled part of the training data and the combination of labeled and unlabeled data. We use the SSS and EM-CBL1 algorithms to train a classifier using both labeled and unlabeled data (we do not search for the structure with just the labeled part because it is too small for performing a full structure search).

Table 10.11 shows the results of the experiments. We see that with NB and TAN, when using 200 and 300 labeled samples, adding the unlabeled data degrades the performance of the classifiers, and we would have been better off not using the unlabeled data. We also see that EM-CBL1 performs poorly in both cases. Using the SSS algorithm, we are able to improve the results and utilize the unlabeled data to achieve performance which is higher than using just the labeled data with NB and TAN. The fact that the performance is lower than in the case when all the training set was labeled (about 75% compared to over 80%) implies that the relative value of labeled data is higher than of unlabeled data, as was shown by Castelli [Castelli, 1994]. However, had there been more unlabeled data, the performance would be expected to improve.

Table 10.11. Classification results for facial expression recognition with labeled and unlabeled data.

Dataset	NB-L	EM-NB	TAN-L	EM-TAN	EM-CBL1	SSS
Cohn-Kanade	272.5±1.4	69.1±1.4	72.9±1.4	69.3±1.4	66.2±1.5	**74.8±1.4**
Chen-Huang	71.3±0.8	58.5±0.8	72.5±0.7	62.9±0.8	65.9±0.8	**75.0±0.7**

5. Discussion

We showed that the TAN and SSS algorithms can be used to enhance the performance of facial expression recognition over the simple Naive Bayes classifier for the person independent and dependent approaches.

Learning the structure of the Bayesian networks also showed that there is a weak dependency between the motion in the lower part of the face and the upper face, an observation that agrees with physiological intuition. The experiments also showed that allowing for learning of the dependencies also enhanced the classification performance with unlabeled data, when using the classification driven SSS algorithm. Such a result suggests that it is not enough to learn first order dependencies between the motion units (as in the case of TAN), rather, more complex dependencies are necessary to truly model the dependencies between facial motions.

Are the recognition rates sufficient for real world use? We think that it depends upon the particular application. In the case of image and video retrieval from large databases, the current recognition rates could aid in finding the right image or video by giving additional options for the queries. For future research, the integration of multiple modalities such as voice analysis and context would be expected to improve the recognition rates and eventually improve the computer's understanding of human emotional states. Voice and gestures are widely believed to play an important role as well [Chen, 2000; De Silva et al., 1997], and physiological states such as heart beat and skin conductivity are being suggested [Cacioppo and Tassinary, 1990]. People also use context as an indicator of the emotional state of a person. The advantage of using Bayesian network classifiers is that they provide a good framework of fusing different modalities in an intuitive and coherent manner. This work is therefore a first step towards building a more comprehensive system of recognizing human's affective state by computers.

Chapter 11

APPLICATION: BAYESIAN NETWORK CLASSIFIERS FOR FACE DETECTION

Images containing faces are essential to intelligent vision-based human computer interaction. To buld fully automated systems that analyze the information contained in face images, robust and efficient face detection algorithms are required. Among the face detection methods, the ones based on learning algorithms have attracted much attention recently and have demonstrated excellent results.

This chapter presents a discussion on semi-supervised learning of probabilistic mixture model classifiers for face detection. Based on our complete theoretical analysis of semi-supervised learning using maximum likelihood presented in Chapter 4 we discuss the possibility of structure learning of Bayesian networks for face detection. We show that learning the structure of Bayesian networks classifiers enables learning of good classifiers for face detection with a small labeled set and a large unlabeled set.

1. Introduction

Many of the recent applications designed for human-computer intelligent interaction applications have used the human face as an input. Systems that perform face tracking for various applications, facial expression recognition and pose estimation of faces all rely on detection of human faces in the video frames [Pentland, 2000]. The rapidly expanding research in face processing is based on the premise that information about user's identity, state, and intend can be extracted from images and that computers can react accordingly, e.g., by observing a person's facial expression. In the last years, face and facial expression recognition have attracted much attention despite the fact that they have been studied for more than 20 years by psychophysicists, neuroscientists, and engineers. Many research demonstrations and commercial applications have been developed from these efforts.

Given an arbitrary image, the goal of face detection is to automatically locate a human face in an image or video, if it is present. Face detection in a general setting is a challenging problem due to the variability in scale, location, orientation (up-right, rotated), and pose (frontal, profile). Facial expression, occlusion, and lighting conditions also change the overall appearance of faces. Yang et al. [Yang et al., 2002] summarize in their comprehensive survey the challenges associated with face detection:

- **Pose.** The images of a face vary due to the relative camera-face pose (frontal, 45 degree, profile, upside down), and some facial features (e.g., an eye or the nose) may become partially or wholly occluded.

- **Presence or absence of structural components.** Facial features such as beards, mustaches, and glasses may or may not be present and there is a great deal of variability among these components including shape, color, and size.

- **Facial expression.** The appearance of faces is directly affected by the facial expression of the persons.

- **Occlusion.** Faces may be partially occluded by other objects. In an image with a group of people, some faces may partially occlude other faces.

- **Image orientation.** Face images directly vary for different rotations about the camera's optical axis.

- **Imaging conditions.** When the image is formed, factors such as lighting (spectra, source distribution and intensity) and camera characteristics (sensor response, lenses) affect the appearance of a face.

There are many closely related problems of face detection. *Face localization* aims to determine the image position of a single face; this is a simplified detection problem with the assumption that an input image contains only one face [Lam and Yan, 1994; Moghaddam and Pentland, 1997]. The goal of *facial feature detection* is to detect the presence and location of features, such as eyes, nose, nostrils, eyebrow, mouth, lips, ears, etc., with the assumption that there is only one face in an image [Craw et al., 1992; Graf et al., 1995]. *Face recognition* or *face identification* compares an input image (probe) against a database (gallery) and reports a match, if any [Chellappa et al., 1995; Samal and Iyengar, 1992; Turk and Pentland, 1991]. The purpose of *face authentication* is to verify the claim of the identity of an individual in an input image [Tefas et al., 1998], while *face tracking* methods continuously estimate the location and possibly the orientation of a face in an image sequence in real time [Edwards et al., 1998; Darell et al., 2000; Crowley and Berard, 1997]. *Facial expression recognition* concerns identifying the affective states (happy, sad, disgusted, etc.) of humans [Pantic and Rothkrantz, 2000; Pantic and Rothkrantz, 2003; Fasel and Luettin, 2003]. Evidently, face detection is the first step in any automated system which solves the above problems. In this chapter, we do not present a complete face detection system. We limit ourselves to present a face detection methodology that can use both labeled and unlabeled data and which can easily be aplied to other face detection methods.

To solve the problem of face detection, four main approaches can be taken:

- **Knowledge-based methods.** These rule-based methods encode human knowledge of what constitutes a typical face. Usually, the rules capture the relationships between facial features. These methods are designed mainly for face localization.

- **Feature invariant approaches.** These algorithms aim to find structural features that exist even when the pose, viewpoint, or lighting conditions vary, and then use the these to locate faces. These methods are designed mainly for face localization.

- **Template matching methods.** Several standard patterns of a face are stored to describe the face as a whole or the facial features separately. The correlations between an input image and the stored patterns are computed for detection. These methods have been used for both face localization and detection.

- **Appearance-based methods.** In contrast to template matching, the models (or templates) are learned from a set of training images which should capture the representative variability of facial appearance. These learned models are then used for detection. These methods are designed mainly for face detection.

2. Related Work

Table 11.1 summarizes algorithms and representative works for face detection in a single image within the four categories presented in the previous section [Yang et al., 2002]. In this chapter, we focus on the appearance based methods.

There have been numerous appearance based approaches. We mention a few from recent years and refer to the detailed reviews of Yang et al. [Yang et al., 2002] and Hjelmas and Low [Hjelmas and Low, 2001] for further details.

While in template matching methods the templates are predefined by experts, the templates in appearance- based methods are learned from examples in images. In general, appearance-based methods rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face and nonface images. The learned characteristics are in the form of distribution models or discriminant functions that are consequently used for face

Table 11.1. Categorization of methods for face detection in a single image.

Approach	Representative works
Knowledge-based	
	Multiresolution rule-based method [Yang and Huang, 1994]
Feature invariant	
- Facial features	Grouping of edges [Leung et al., 1995; Yow and Cipolla, 1997]
- Texture	Space Gray-Level Dependence matrix (SGLD) of face patterns [Dai and Nakano, 1996]
- Skin Color	Mixture of Gaussian [Yang and Waibel, 1996; McKenna et al., 1998]
- Multiple Features	Integration of skin color, size, and shape [Kjeldsen and Kender, 1996]
Template matching	
- Predefined face templates	Shape template [Craw et al., 1992]
- Deformable templates	Active Shape Model (ASM) [Lanitis et al., 1995a]
Appearance-based method	
- Eigenface	Eigenvector decomposition and clustering [Turk and Pentland, 1991]
- Distribution-based	Gaussian distribution and multilayer perceptron [Sung and Poggio, 1998]
- Neural Network	Ensemble of neural networks and arbitration schemes [Rowley et al., 1998a; Kouzani, 2003]
- Support Vector Machine (SVM)	SVM with polynomial kernel [Osuna et al., 1997]
- Naive Bayes classifier	Joint statistics of local appearance and position [Schneiderman and Kanade, 1998]
- Hidden Markov Model (HMM)	Higher order statistics with HMM [Rajagalopan et al., 1998]
- Information-theoretical approach	Kullback relative information [Colmenarez and Huang, 1997; Lew, 1996]

detection. Meanwhile, dimensionality reduction is usually carried out for the sake of computation efficiency and detection efficacy. Many appearance-based methods can be understood in a probabilistic framework. An image or feature vector derived from an image is viewed as a random variable x, and this random variable is characterized for faces and nonfaces by the class-conditional density functions $p(x|\mathit{face})$ and $p(x|\mathit{nonface})$. Bayesian classification or maximum likelihood can be used to classify a candidate image location as face or nonface. Unfortunately, a straightforward implementation of Bayesian classification is infeasible due to the high dimensionality of x, and because $p(x|\mathit{face})$ and $p(x|\mathit{nonface})$ are multimodal. Another problem is the fact that it is not yet understood if there are natural parameterized forms for $p(x|\mathit{face})$ and $p(x|\mathit{nonface})$. As a consequence, much of the work in an appearance-based method concerns empirically validated parametric and nonpara-metric approximations to $p(x|\mathit{face})$ and $p(x|\mathit{nonface})$.

Another approach in appearance-based methods is to find a discriminant function (i.e., decision surface, separating hyperplane, threshold function) between face and nonface classes. Conventionally, image patterns are projected to a lower dimensional space and then a discriminant function is formed (usually based on distance metrics) for classification [Turk and Pentland, 1991], or a nonlinear decision surface can be formed using multilayer neural networks [Rowley et al., 1998a; Kouzani, 2003]. Recently, support vector machines and other kernel methods have been proposed. These methods implicitly project patterns to a higher dimensional space and then form a decision surface between the projected face and nonface patterns [Osuna et al., 1997].

Turk and Pentland applied principal component analysis to face recognition and detection [Turk and Pentland, 1991]. Similar to [Kirby and Sirovich, 1990], principal component analysis on a training set of face images is performed to generate *eigenfaces* which span a subspace (called the face space) of the image space. Images of faces are projected onto the subspace and clustered. Similarly, nonface training images are projected onto the same subspace and clustered. The idea is that the images of faces do not change radically when projected onto the face space, while the projection of nonface images appear quite different. To detect the presence of a face in a scene, the distance between an image region and the face space is computed for all locations in the image. The distance from face space is used as a measure of *faceness*, and the result of calculating the distance from face space is a *face map*. A face can then be detected from the local minima of the face map.

Sung and Poggio [Sung and Poggio, 1998] developed a distribution-based system for face detection which demonstrated how the distributions of image patterns from one object class can be learned from positive and negative examples (i.e., images) of that class. Their system consists of two components, distribution-based models for face/nonface patterns and a multilayer perceptron classifier. A probabilistic visual learning method based on density estimation in a high-dimensional space using an eigenspace decomposition was developed by Moghaddam and Pentland [Moghaddam and Pentland, 1997]. Principal component analysis (PCA) is used to define the subspace best representing a set of face patterns. This method decomposes the vector space into two mutually exclusive and complementary subspaces: the principal subspace (or feature space) and its orthogonal complement. Therefore, the target density is decomposed into two components: the density in the principal subspace (spanned by the principal components) and its orthogonal complement (which is discarded in standard PCA). A multivariate Gaussian and a mixture of Gaussians are used to learn the statistics of the local features of a face. These probability densities are then used for object detection based on maximum likelihood estimation.

Rowley et al. [Rowley et al., 1998a] and Kouzani [Kouzani, 2003] used Neural networks to detect faces in images by training from a corpus of face and nonface images. A multilayer neural network is used to learn the face and nonface patterns from face/nonface images (i.e., the intensities and spatial relationships of pixels). One limitation of the method of Rowley et al. [Rowley et al., 1998a] is that it can only detect upright, frontal faces. To address this problem, Rowley et al. [Rowley et al., 1998b] extended this method to detect rotated faces using a router network which processes each input window to determine the possible face orientation and then rotates the window to a canonical orientation. However, the new system has a lower detection rate on upright faces than the upright detector. Nevertheless, the system is able to detect 76.9 percent of faces over two large test sets with a small number of false positives.

In contrast to the methods in [Osuna et al., 1997] and [Sung and Poggio, 1998] which model the global appearance of a face, Schneiderman and Kanade [Schneiderman and Kanade, 1998] described a naive Bayes classifier to estimate the joint probability of local appearance and position of face patterns (subregions of the face) at multiple resolutions. They emphasize local appearance because some local patterns of an object are more unique than others; the intensity patterns around the eyes are much more distinctive than the pattern found around the cheeks. There are two reasons for using a naive Bayes classifier (i.e., no statistical dependency between the subregions). First, it provides better estimation of the conditional density functions of these subregions. Second, a naive Bayes classifier provides a functional form of the posterior probability to capture the joint statistics of local appearance and position on the object. At each scale, a face image is decomposed into four rectangular subregions. These subregions are then projected to a lower dimensional space using PCA and quantized into a finite set of patterns, and the statistics of each projected subregion are estimated from the projected samples to encode local appearance. Under this formulation, their method decides that a face is present when the likelihood ratio is larger than the ratio of prior probabilities.

Yang et al. [Yang et al., 2000] used SNoW based classifiers to learn the face and non-face discrimination boundary on natural face images. They detect faces with different features and expressions, in different poses, and under different lighting conditions.

Colmenarez and Huang [Colmenarez and Huang, 1997] used maximum entropic discrimination between faces and non-faces to perform maximum likelihood classification, which was used for a real time face tracking system. Images from the training set of each class (i.e., face and nonface class) are analyzed as observations of a random process and are characterized by two probability functions. The authors used a family of discrete Markov processes to model the face and background patterns and to estimate the probability model. The learning process is converted into an optimization problem to select the Markov process that maximizes the information-based discrimination between the two classes. The likelihood ratio is computed using the trained probability model and used to detect the faces. Wang et al. [Wang et al., 2002] learned a minimum spanning weighted tree for learning pairwise dependencies graphs of facial pixels, followed by a discriminant projection to reduce complexity. Viola and Jones [Viola and Jones, 2004] used boosting and a cascade of classifiers for face detection.

3. Applying Bayesian Network Classifiers to Face Detection

Among the different works mentioned above, those of Colmenarez and Huang [Colmenarez and Huang, 1997] and Wang et al. [Wang et al., 2002] are more related to the Bayesian network classification methods. Both learn some ‘structure’ between the facial pixels and combine them to a probabilistic classification rule. Both use the entropy between the different pixels to learn pairwise dependencies.

Face detection provides interesting challenges to the underlying pattern classification and learning techniques. When a raw or filtered image is considered as an input to a pattern classifier, the dimesion of the space is extremely large (i.e., the number of pixels in the normalized training images). The classes of face and non-face images are decidely characterized by multimodal distribution functions and effective decision boundaries are likely to be non-linear in the image space. To be effective, the classifiers must be able to extrapolate from a modest number of training samples.

In the following, we propose to use Bayesian network classifiers, with the image pixels of a predefined window size as the features in the Bayesian network. We propose to use the TAN classifier to learn dependencies between the features, and show that using the stochastic structure search algorithm provides further enhancement of the classification result. The aim of this work is to demonstrate the ability of Bayesian network classifiers to learn appearance based face models for the face detection problem with both labeled and unlabeled data.

Our approach in detecting faces is an appearance based approach, where the intensity of image pixels serve as the features for the classifier. In a natural image, faces can appear at different scales, rotations and location. For learning and defining the Bayesian network classifiers, we must look at fixed size windows and learn how a face appears in such as windows, where we assume that the face appears in most of the window's pixels.

The goal of the classifier would be to determine if the pixels in a fixed size window are those of a face or non-face. While faces are a well defined concept, and have a relatively regular appearance, it is harder to characterize non-faces. We therefore model the pixel intensities as discrete random variables, as it would be impossible to define a parametric probability distribution function (pdf) for non-face images. For 8-bit representation of pixel intensity, each pixel has 256 values. Clearly, if all these values are used for the classifier, the number of parameters of the joint distribution is too large for learning dependencies between the pixels (as in the case of TAN classifiers). Therefore, there is a need to reduce the number of values representing pixel intensity. Colmenarez and Huang [Colmenarez and Huang, 1997] used 4 values per pixel using fixed and equal bin sizes. We use non-uniform discretization using the class conditional entropy as the mean to bin the 256 values to a smaller number. We use the MLC++ software for that purpose as described in [Dougherty et al., 1995].

Note that our methodology can be extended to other face detection methods which use different features. The complexity of our method is $O(n)$, where n is the number of features (pixels in our case) considered in each image window.

4. Experiments

We test the different approaches described in Chapter 7, with both labeled data and unlabeled data. For training the classifier we used a dataset consisting of 2,429 faces and 10,000 non faces obtained from the MIT CBCL Face database #1 [MITFaceDB, 2000]. Examples of face images from the database are presented in Figure 11.1. Each face image is cropped and resampled to a 19×19 window, thus we have a classifier with 361 features. We also randomly rotate and translate the face images to create a training set of 10,000 face images. In addition we have available 10,000 non-face images. We leave out $1,000$ images (faces and non-faces) for testing and train the Bayesian network classifier on the remaining 19,000. In all the experiments we learn a Naive Bayes, a TAN, and a general generative Bayesian network classifier, the latter using the SSS algorithm.

To compare the results of the classifiers, we use the receiving operating characteristic (ROC) curves. The ROC curves show, under different classification thresholds, ranging from 0 to 1, the probability of detecting a face in a face image, $P_D = P(\hat{C} = face|C = face)$, against the probability of falsely detecting a face in a non-face image,$P_{FD} = P(\hat{C} = face|C \neq face)$.

We first learn using all the training data being labeled (that is 19,000 labeled images). Figure 11.2 shows the resultant ROC curve for this case. The classifier learned with the SSS algorithm outperforms both TAN and NB classifiers,

Figure 11.1. Randomly selected face examples.

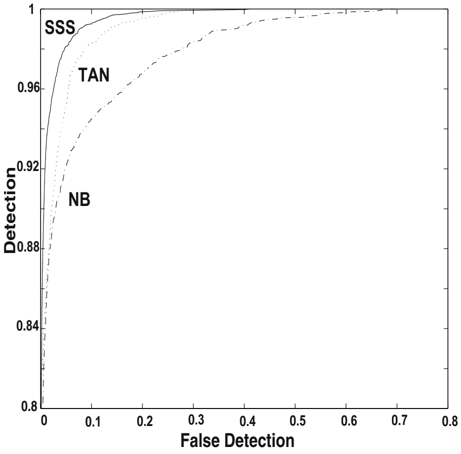


Figure 11.2. ROC curves showing detection rates of faces compared to false detection of faces of the different classifiers (NB, TAN, SSS) when all the data are labeled (no unlabeled data).

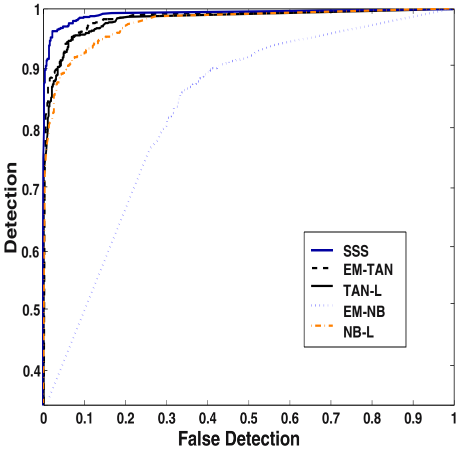


Figure 11.3. ROC curves showing detection rates of faces compared to false detection of faces with 97.5% unlabeled data for different classifiers: SSS, Naive Bayes learned with labeled data only (NB-L) and with labeled and unlabeled data (EM-NB), and TAN learned with labeled data only (TAN-L) and with labeled and unlabeled data (EM-TAN).

and all perform quite well, achieving about 98% detection rates with a low rate of false alarm.

Next, we remove the labels of some of the training data and train the classifiers. In the first case presented in Figure 11.3, we removed the labeles of 97.5% of the training data (leaving only 475 labeled images) and train the classifiers. We see that the NB classifier using both labeled and unlabeled data (EM-NB) performs very poorly. The TAN based only on the 475 labeled images (TAN-L) and the TAN based on the labeled and unlabeled images (EM-TAN) are close in performance, thus there was no significant degradation of performance when adding the unlabeled data. The classifier based on the EM-CBL1 algorithm has the fastest increase in detection rate compared to false alarms, but at some point the increase becomes slow which leads to an inferion performance compared to the classifier trained with SSS.

Figure 11.4 shows the ROC curve with only 250 labeled data used (the labels of about 98.7% of the training data were removed). Again, NB with both labeled and unlabeled data (EM-NB) performs poorly, while SSS outperforms the other classifiers with no great reduction of performance compared to the ROC curves presented in Figure 11.3. Note that the ROC curves in Figures 11.4 and 11.3 corresponding to the classifiers trained only with a small amount of labeled data are significantly lower than the ones in Figure 11.2. However, when both labeled and unlabeled data are used the curves are only marginally

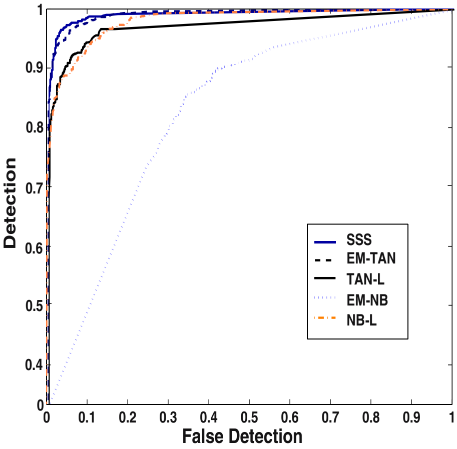


Figure 11.4. ROC curves showing detection rates of faces compared to false detection of faces with 98.7% unlabeled data for different classifiers: SSS, Naive Bayes learned with labeled data only (NB-L) and with labeled and unlabeled data (EM-NB), and TAN learned with labeled data only (TAN-L) and with labeled and unlabeled data (EM-TAN).

lower. The experiment shows that using structure search, the unlabeled data was utilized successfully to achieve a classifier almost as good as if all the data was labeled.

In Table 11.2 we summarize the results obtained for different algorithms and in the presence of increasing number of unlabeled data. We fixed the false alarm to 1%, 5%, and 10% and we computed the detection rates. Note that the detection rates for NB are lower than the ones obtained for the other detectors. For benchmarking we implemented a SVM classifier (we used the implementation of Osuna et al. [Osuna et al., 1997]). Note that this classifier starts off very good, but does not improve performance. Overall, the results obtained with SSS are the best. We see that even in the most difficult cases, there was sufficient amount of unlabeled data to achieve almost the same performance as with a large sized labeled dataset.

We also tested our system on the CMU test set [Rowley et al., 1998a] consisting of 130 images with a total of 507 frontal faces. The results are summarized in Table 11.3. Note that we obtained comparable results with the results obtained by Viola and Jones [Viola and Jones, 2004] and better than the results of Rowley et al. [Rowley et al., 1998a]. Examples of the detection results on some of the images of the CMU test are presented in Figure 11.5. We noticed similar failure modes as Viola and Jones [Viola and Jones, 2004]. Since, the face detector was trained only on frontal faces our system failes to detect

Table 11.2. Detection rates (%) for various numbers of false positives.

False positives Detector		1%	5%	10%
NB	19,000 labeled	74.31	89.21	92.72
	475 labeled	68.37	86.55	89.45
	475 labeled + 18,525 unlabeled	66.05	85.73	86.98
	250 labeled	65.59	84.13	87.67
	250 labeled + 18,750 unlabeled	65.15	83.81	86.07
TAN	19,000 labeled	91.82	96.42	99.11
	475 labeled	86.59	90.84	94.67
	475 labeled + 18,525 unlabeled	85.77	90.87	94.21
	250 labeled	75.37	87.97	92.56
	250 labeled + 18,750 unlabeled	77.19	89.08	91.42
SSS	19,000 labeled	90.27	98.26	99.87
	475 labeled + 18,525 unlabeled	88.66	96.89	98.77
	250 labeled + 18,750 unlabeled	86.64	95.29	97.93
SVM	19,000 labeled	87.78	93.84	94.14
	475 labeled	82.61	89.66	91.12
	250 labeled	77.64	87.17	89.16

faces if they have a significant rotation out of the plane (toward a profile view). The detector has also problems with the images in which the faces appear dark and the background is relatively light. Inevitably, we also detect false positive especially in some texture regions.

Table 11.3. Detection rates (%) for various numbers of false positives on the CMU test set.

False positives Detector		10%	20%
SSS	19,000 labeled	91.7	92.84
	475 labeled + 18,525 unlabeled	89.67	91.03
	250 labeled + 18,750 unlabeled	86.64	89.17
Viola-Jones [Viola and Jones, 2004]		92.1	93.2
Rowley et al. [Rowley et al., 1998a]		-	89.2

5. Discussion

In this chapter, we suggested a methodology for learning to detect faces using both labeled and unlabeled data samples. Consequently, we applied the algorithms presented in Chapter 7 for learning the structure of a Bayesian network classifier to detect faces in images. We humbly note that while finding

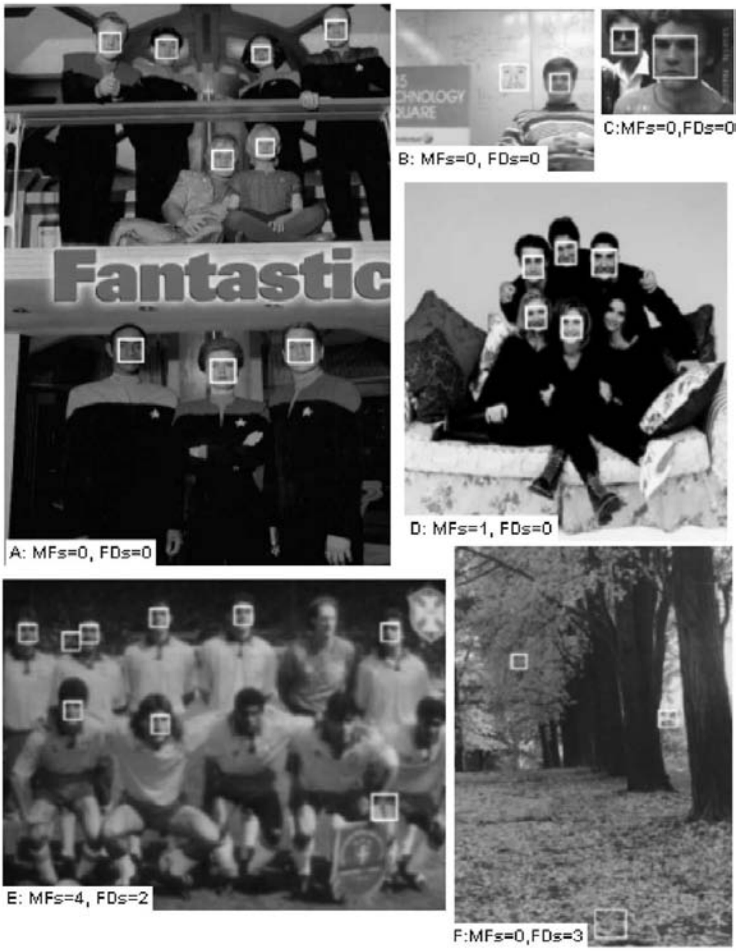


Figure 11.5. Output of the system on some images of the CMU test using the SSS classifier learned with 19,000 labeled data. MFs represents the number of missed faces and FDs is the number of false detections.

a good classifier is a major part of any face detection system, there are many more components that need to be designed for such a system to work on natural images (e.g., ability to detect at multi-scales, highly varying illumination, large rotations of faces and partial occlusions).

Another issue which was artificially solved in our case is the problem of determining the prior of the class variable. Can we really define the prior probability of a face being in an image? Probably not. However, we need to define, even artificially, such prior probability if we want to use maximum a-posteriori (MAP) classification. Of course, the prior probability will be determined by our training set, but the assumptions that the samples are randomly sampled from the joint distribution does not hold. We fully control the ratio between face and non-face images. A solution to this problem is done by the use of the Neiman-Pearson ratio - we determine the allowable false alarm rate, and we can set the classification threshold different than $1/2$ so we achieve this rate. A good classifier will achieve a good face detection rate. Our experiments do show that the ROC curve of our classifiers can achieve high detection of faces with relatively low false alarms (albeit not zero).

References

Achlioptas, D. (2001). Database-friendly random projections. In *Symposium on Principles of Database Systems*, pages 274–281.

Ahmed, S.W. and Lachenbruch, P.A. (1977). Discriminant analysis when scale contamination is present in the initial sample. In *Classification and Clustering*, pages 331–353, New York. Academic Press Inc.

Arriaga, R.I. and Vempala, S. (1999). An algorithmic theory of learning: Robust concepts and random projection. In *Proc. Foundations of Computer Science*, pages 616–623.

Bahl, L., Brown, P., de Souza, P., and Mercer, R. (1993). Estimating hidden Markov model parameters so as to maximize speech recognition accuracy. *IEEE Transactions on Speech and Audio Processing*, 1:77–83.

Balasubramanian, V., Myung, J., and Pitt, M.A. (2000). Counting probability distributions: Differential geometry and model selection. In *Proceedings of National Academy of Sciences*, volume 97, pages 11170–11175.

Baluja, S. (1998). Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data. In *Neural Information and Processing Systems*, pages 854–860.

Bassili, J.N. (1979). Emotion recognition: The role of facial movement and the relative importance of upper and lower areas of the face. *Journal of Personality and Social Psychology*, 37(11):2049–2058.

Bengio, Y. and Frasconi, P. (1996). Input-output HMM's for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249.

Bennett, K. and Demiriz, A. (1998). Semi-supervised support vector machines. In *Neural Information and Processing Systems*, pages 368–374.

Berk, R.H. (1966). Limiting behavior of posterior distributions when the model is incorrect. *Annals of Mathematical Statistics*, 37:51–58.

Bianchi-Berthouze, N. and Lisetti, C. (2002). Modeling multimodal expression of user's affective subjective experience. *User Modeling and User-Adapted Interaction*, 12:49–84.

Binder, J., Koller, D., Russell, S.J., and Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2):213–244.

Bischof, W. and Caelli, T. (2001). Learning spatio-temporal relational structures. *Applied Artificial Intelligence*, 15(8):707–722.

Black, M.J. and Yacoob, Y. (1995). Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *Proc. International Conference on Computer Vision*, pages 374–381.

Blake, C.L. and Merz, C.J. (1998). UCI repository of machine learning databases.

Blockeel, H. and De Raedt, L. (1998). Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Conference on Learning Theory*, pages 92–100.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M.K. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–865.

Brand, M. (1998). An entropic estimator for structure discovery. In *Neural Information and Processing Systems*, pages 723–729.

Brand, M. and Kettnaker, V. (2000). Discovery and segmentation of activities in video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):844–851.

Brand, M., Oliver, N., and Pentland, A. (1997). Coupled hidden Markov models for complex action recognition. In *International Conference on Pattern Recognition*, pages 994–999.

Brandstein, M.S. and Silverman, H.F. (1997). A practical methodology for speech source localization with microphone arrays. *Computer, Speech, and Language*, 1(2):91–126.

Bruce, R. (2001). Semi-supervised learning using prior probabilities and EM. In *International Joint Conference on Artificial Intelligence, Workshop on Text Learning: Beyond Supervision*.

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

Buxton, H. and Gong, S. (1995). Advanced visual surveillance using Bayesian networks. In *International Conference on Computer Vision*, pages 111–123.

Cacioppo, J.T. and Tassinary, L.G. (1990). Inferring psychological significance from physiological signals. *American Psychologist*, 45:16–28.

Cannon, W. B. (1927). The James-Lange theory of emotion: A critical examination and an alternative theory. *American Journal of Psychology*, 39:106–124.

Castelli, V. (1994). *The Relative Value of Labeled and Unlabeled Samples in Pattern Recognition*. PhD thesis, Stanford University.

Castelli, V. and Cover, T. (1995). On the exponential value of labeled samples. *Pattern Recognition Letters*, 16:105–111.

Castelli, V. and Cover, T. (1996). The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter. *IEEE Transactions on Information Theory*, 42(6):2102–2117.

Chellappa, R., Wilson, C.L., and Sirohey, S. (1995). Human and machine recognition of faces: A survey. *Proceedings of the IEEE*, 83(5):705–740.

Chen, L.S. (2000). *Joint Processing of Audio-visual Information for the Recognition of Emotional Expressions in Human-computer Interaction*. PhD thesis, University of Illinois at Urbana-Champaign, Dept. of Electrical Engineering.

Chen, T. and Rao, R. (1998). Audio-visual integration in multimodal communication. *Proceedings of the IEEE*, 86(5):837–852.

Cheng, J., Bell, D.A., and Liu, W. (1997). Learning belief networks from data: An information theory based approach. In *International Conference on Information and Knowledge Management*, pages 325–331.

Cheng, J. and Greiner, R. (1999). Comparing Bayesian network classifiers. In *Proc. Conference on Uncertainty in Artificial Intelligence*, pages 101–107.

Chhikara, R. and McKeon, J. (1984). Linear discriminant analysis with misallocation in training samples. *Journal of the American Statistical Association*, 79:899–906.

Chittineni, C. (1981). Learning with imperfectly labeled examples. *Pattern Recognition*, 12:271–281.

Chow, C.K. and Liu, C.N. (1968). Approximating discrete probability distribution with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467.

Christian, A.D. and Avery, B.L. (1998). Digital Smart Kiosk project. In *ACM SIGCHI*, pages 155–162.

Clarkson, B. and Pentland, A. (1999). Unsupervised clustering of ambulatory audio and video. In *International Conference on Accoustics, Speech, and Signal Processing*, pages 3037–3040.

Cohen, I., Cozman, F.G., and Bronstein, A. (2002a). On the value of unlabeled data in semi-supervised learning based on maximum-likelihood estimation. Technical Report HPL-2002-140, HP-Labs.

Cohen, I., Garg, A., and Huang, T.S. (2000). Emotion recognition using multi-level HMMs. In *Neural Information Processing Systems, Workshop on Affective Computing*.

Cohen, I., Sebe, N., Cozman, F., Cirelo, M., and Huang, T.S. (to appear, 2004). Semi-supervised learning of classifiers: Theory, algorithms, and applications to human-computer interaction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Cohen, I., Sebe, N., Cozman, F.G., Cirelo, M.C., and Huang, T.S. (2003a). Learning Bayesian network classifiers for facial expression recognition using both labeled and unlabeled data. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 595–601.

Cohen, I., Sebe, N., Garg, A., Chen, L., and Huang, T.S. (2003b). Facial expression recognition from video sequences: Temporal and static modeling. *Computer Vision and Image Understanding*, 91(1-2):160–187.

Cohen, I., Sebe, N., Garg, A., and Huang, T.S. (2002b). Facial expression recognition from video sequences. In *International Conference on Multimedia and Expo*, volume 2, pages 121–124.

Cohen, I., Sebe, N., Sun, Y., Lew, M.S., and Huang, T.S. (2003c). Evaluation of expression recognition techniques. In *International Conference on Image and Video Retrieval*, pages 184–195.

Collins, M. and Singer, Y. (2000). Unupervised models for named entity classification. In *International Conference on Machine Learning*, pages 327–334.

Colmenarez, A.J. and Huang, T.S. (1997). Face detection with information based maximum discrimination. In *Proc. IEEE Conference on Computer Vision and Pattern Recogntion*, pages 782–787.

Comite, F. De, Denis, F., Gilleron, R., and Letouzey, F. (1999). Positive and unlabeled examples help learning. In *Proc. International Conference on Algorithmic Learning Theory*, pages 219–230.

Cooper, D.B. and Freeman, J.H. (1970). On the asymptotic improvement in the outcome of supervised learning provided by additional nonsupervised learning. *IEEE Transactions on Computers*, C-19(11):1055–1063.

Cooper, G. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:308–347.

Corduneanu, A. and Jaakkola, T. (2002). Continuations methods for mixing heterogeneous sources. In *Proc. Conference on Uncertainty in Artificial Intelligence*, pages 111–118.

Cormen, T.H., Leiserson, C.E., and Rivest, R.L. (1990). *Introduction to Algorithms*. MIT Press, Cambridge, MA.

Cover, T.M. and Thomas, J.A. (1991). *Elements of Information Theory*. John Wiley and Sons, New York.

Cozman, F.G. and Cohen, I. (2001). Unlabeled data can degrade classification performance of generative classifiers. Technical Report HPL-2001-234, HP-Labs.

Cozman, F.G. and Cohen, I. (2003). The effect of modeling errors in semi-supervised learning of mixture models: How unlabeled data can degrade performance of generative classifiers. Technical report, http://www.poli.usp.br/p/fabio.cozman/ Publications/lul.ps.gz.

Craw, I., Tock, D., and Bennett, A. (1992). Finding face features. In *European Conference on Computer Vision*, pages 92–96.

Crowley, J. and Berard, F. (1997). Multi-modal tracking of faces for video communications. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 640–645.

Dai, Y. and Nakano, Y. (1996). Face-texture model based on SGLD and its application in face detection in color scene. *Patern Recognition*, 29(6):1007–1017.

Darell, T., Gordon, G., Harville, M., and Woodfill, J. (2000). Integrated person tracking using stereo, color, and pattern decision. *International Journal of Computer Vision*, 37(2):175–185.

Darwin, C. (1890). *The Expression of the Emotions in Man and Animals*. John Murray, London, 2nd edition.

Dawid, A.P. (1976). Properties of diagnostic data distributions. *Biometrics*, 32:647–658.

De Silva, L.C., Miyasato, T., and Natatsu, R. (1997). Facial emotion recognition using multimodal information. In *Proc. IEEE International Conference on Information, Communications, and Signal Processing*, pages 397–401.

DeJong, K. (1988). Learning with genetic algorithms: An overview. *Machine Learning*, 3:121–138.

Dempster, A.P., Laird, N.M., and Rubin, D.B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Devroye, L., Gyorfi, L., and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. Springer Verlag, New York.

Domingos, P. and Pazzani, M. (1997). Beyond independence: Conditions for the optimality of the simple Bayesian classifier. *Machine Learning*, 29:103–130.

Dougherty, J., Kohavi, R., and Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *International Conference on Machine Learning*, pages 194–202.

Duda, R.O. and Hart, P.E. (1973). *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York.

Edwards, G., Taylor, C., and Cootes, T. (1998). Learning to identify and track faces in image sequences. In *Proc. International Conference on Computer Vision*, pages 317–322.

Efron, B. and Tibshirani, R.J. (1993). *An Introduction to the Bootstrap*. Chapman Hall, New York.

Ekman, P., editor (1982). *Emotion in the Human Face*. Cambridge University Press, New York, NY, 2nd edition.

Ekman, P. (1994). Strong evidence for universals in facial expressions: A reply to Russell's mistaken critique. *Psychological Bulletin*, 115(2):268–287.

Ekman, P. and Friesen, W.V. (1978). *Facial Action Coding System: Investigator's Guide*. Consulting Psychologists Press.

Elkan, C. (1997). Boosting and naive Bayesian learning. Technical Report CS97-557, University of California, San Diego.

Esposito, F. and Malerba, D. (2001). Machine learning in computer vision. *Applied Artificial Intelligence*, 15(8).

Essa, I.A. and Pentland, A.P. (1997). Coding, analysis, interpretation, and recognition of facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):757–763.

Faigin, G. (1990). *The Artist's Complete Guide To Facial Expression*. Watson-Guptill Publications, New York, NY.

Fasel, B. and Luettin, J. (2003). Automatic facial expression analysis: A survey. *Pattern Recognition*, 36:259–275.

Feder, M. and Merhav, N. (1994). Relation between entropy and error probability. *IEEE Transactions on Information Theory*, 40:259–266.

Fernyhough, J., Cohn, A., and Hogg, D. (1998). Building qualitative event models automatically from visual input. In *International Conference on Computer Vision*, pages 350–355.

Fine, S., Singer, Y., and Tishby, N. (1998). The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, 32(1):41–62.

Forbes, J., Huang, T., Kanazawa, K., and Russell, S.J. (1995). The BATmobile: Towards a Bayesian automated taxi. In *Proc. International Joint Conference on Artificial Intelligence*, pages 1878–1885.

Friedman, J.H. (1997). On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77.

Friedman, N. (1998). The Bayesian structural EM algorithm. In *Proc. Conference on Uncertainty in Artificial Intelligence*, pages 129–138.

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29(2):131–163.

Friedman, N. and Koller, D. (2000). Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. In *Proc. Conference on Uncertainty in Artificial Intelligence*, pages 201–210.

Galata, A., Johnson, N., and Hogg, D. (2001). Learning variable length Markov models of behaviour. *International Journal of Computer Vision*, 19:398–413.

Ganesalingam, S. and McLachlan, G.J. (1978). The efficiency of a linear discriminant function based on unclassified initial samples. *Biometrika*, 65:658–662.

Garg, A., Pavlovic, V., and Rehg, J. (2003). Boosted learning in dynamic Bayesian networks for multimodal speaker detection. *Proceedings of the IEEE*, 91(9):1355–1369.

Garg, A., Pavlovic, V., Rehg, J., and Huang, T.S. (2000a). Audio–visual speaker detection using dynamic Bayesian networks. In *Proc. International Conference on Automatic Face and Gesture Recognition*, pages 374–471.

Garg, A., Pavlovic, V., Rehg, J., and Huang, T.S. (2000b). Integrated audio/visual speaker detection using dynamic Bayesian networks. In *IEEE Conference on Automatic Face and Gesture Recognition*.

Garg, A. and Roth, D. (2001a). Learning coherent concepts. In *International Workshop on Algorithmic Learning Theory*, pages 135–150.

Garg, A. and Roth, D. (2001b). Understanding probabilistic classifiers. In *European Conference on Machine Learning*, pages 179–191.

Ghahramani, Z. and Jordan, M.I. (1996). Factorial hidden Markov models. *Advances in Neural Information Processing Systems*, 8:472–478.

Ghahramani, Z. and Jordan, M.I. (1997). Factorial hidden Markov models. *Machine Learning*, 29:245–273.

Ghani, R. (2002). Combining labeled and unlabeled data for multiclass text categorization. In *International Conference on Machine Learning*, pages 187–194.

Golding, A.R. (1995). A Bayesian hybrid method for context-sensitive spelling correction. In *Workshop on Very Large Corpora*, pages 39–53.

Golding, A.R. and Roth, D. (1999). A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, 34:107–130.

Goldman, S. and Zhou, Y. (2000). Enhancing supervised learning with unlabeled data. In *International Conference on Machine Learning*, pages 327–334.

Goleman, D. (1995). *Emotional Intelligence*. Bantam Books, New York.

Graf, H., Chen, T., Petajan, E., and Cosatto, E. (1995). Locating faces and facial parts. In *International Workshop Automatic Face and Gesture Recognition*, pages 41–46.

Greiner, R. and Zhou, W. (2002). Structural extension to logistic regression: discriminative parameter learning of belief net classifiers. In *Proc. Annual National Conference on Artificial Intelligence (AAAI)*, pages 167–173.

Hajek, B. (1988). Cooling schedules for optimal annealing. *Mathematics of Operational Research*, 13:311–329.

Haralick, R. and Shapiro, L. (1979). The consistent labeling problem: Part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):173–184.

Herbrich, R. and Graepel, T. (2001). A PAC-Bayesian margin bound for linear classifiers: Why SVMs work. In *Advances in Neural Information Processing Systems*, pages 224–230.

Hilgard, E., Atkinson, R.C., and Hilgard, R.L. (1971). *Introduction to Psychology*. Harcourt Brace Jovanovich, New York, NY, 5th edition.

Hjelmas, E. and Low, B. (2001). Face detection: A survey. *Computer Vision and Image Understanding*, 83(3):236–274.

Hoey, J. (2001). Hierarchical unsupervised learning of facial expression categories. In *Iternational Conference on Computer Vision, Workshop on Detection and Recognition of Events in Video*, pages 99–106.

Hongeng, S., Bremond, F., and Nevatia, R. (2000). Representation and optimal recognition of human activities. In *International Conference on Computer Vision*, volume 1, pages 1818–1825.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K. (1998). The Lumiȩre project: Bayesian user modeling for inferring the goals and needs of software users. In *Conference on Uncertainty in Artificial Intelligence*, pages 256–265.

Horvitz, E., Jacobs, A., and Hovel, D. (1999). Attention-sensitive alerting. In *Conference on Uncertanity in Artificial Intelligence*, pages 305–313.

Hosmer, D.W. (1973). A comparison of iterative maximum likelihood estimates of the parameters of a mixture of two normal distributions under three different types of sample. *Biometrics*, 29:761–770.

Huber, P.J. (1967). The behavior of maximum likelihood estimates under nonstandard conditions. In *Proceedings of the Fifth Berkeley Symposium in Mathematical Statistics and Probability*, pages 221–233.

Huijsman, D.P. and Sebe, N. (to appear, 2004). How to complete performance graphs in content-based image retrieval: Add generality and normalize scope. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Indyk, P. (2001). Algorithmic applications of low-distortion geometric embeddings. In *Foundations of Computer Science*, pages 10 – 31.

Intille, S.S. and Bobick, A.F. (1999). A framework for recognizing multi-agent action from visual evidence. In *National Conference on Artificial Intelligence*, pages 518–525.

Ivanov, Y. and Bobick, A. (2000). Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):852–872.

Izard, C.E. (1994). Innate and universal facial expressions: Evidence from developmental and cross-cultural research. *Psychological Bulletin*, 115(2):288–299.

James, G.M. (2003). Variance and bias for general loss functions. *Machine Learning*, 51:115–135.

James, W. (1890). *The Principles of Psychology*. Henry Holt, New York, NY.

Jebara, T. and Pentland, A. (1998). Maximum conditional likelihood via bound maximization and the CEM algorithm. In *Advances in Neural Information Processing Systems*, pages 494–500.

Jenkins, J.M., Oatley, K., and Stein, N.L., editors (1998). *Human Emotions: A Reader*. Blackwell Publishers, Malden, MA.

Johnson, B. and Greenberg, S. (1999). Judging people's availability for interaction from video snapshots. In *Hawaii International Conference on System Sciences*.

Johnson, W.B. and Lindenstrauss, J. (1984). Extensions of Lipschitz mappings into a Hilbert space. In *Conference in Modern Analysis and Probability*, pages 189–206.

Jordan, M.I., Ghahramani, Z., and Saul, L.K. (1997). Hidden Markov decision trees. In *Advances in Neural Information Processing Systems*.

Kanade, T., Cohn, J., and Tian, Y. (2000). Comprehensive database for facial expression analysis. In *International Conference on Automatic Face and Gesture Recognition*, pages 46–53.

Kay, R.M. (1990). *Entropy and Information Theory*. Springer-Verlag.

Kearns, M., Mansour, Y., Ng, A.Y., and Ron, D. (1997). An experimental and theoretical comparison of model selection methods. *Machine Learning*, 279:7–50.

Kearns, M. and Schapire, R. (1994). Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, 48:464–497.

Kirby, M. and Sirovich, L. (1990). Application of the Karhunen-Loeve procedure for characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):103–108.

Kjeldsen, R. and Kender, J. (1996). Finding skin in color images. In *International Conference on Automatic Face and Gesture Recognition*, pages 312–317.

Kohavi, R. (1996). Scaling up the accuracy of naive Bayes classifiers: A decision-tree hybrid. In *Proc. International Conference on Knowledge Discovery and Data Mining*, pages 194–202.

Kouzani, A.Z. (2003). Locating human faces within images. *Computer Vision and Image Understanding*, 91(3):247–279.

Krishnan, T. and Nandy, S. (1990a). Efficiency of discriminant analysis when initial samples are classified stochastically. *Pattern Recognition*, 23(5):529–537.

Krishnan, T. and Nandy, S. (1990b). Efficiency of logistic-normal supervision. *Pattern Recognition*, 23(11):1275–1279.

Kullback, S. (1968). Probability densities with given marginals. *Annals of Mathematical Statistics*, 39(4):1236–1243.

Lam, K. and Yan, H. (1994). Fast algorithm for locating head boundaries. *Journal of Electronic Imaging*, 3(4):351–359.

Lang, P. (1995). The emotion probe: Studies of motivation and attention. *American Psychologist*, 50(5):372–385.

Lanitis, A., Taylor, C.J., and Cootes, T.F. (1995a). An automatic face identification system using flexible appearance models. *Image and Vision Computing*, 13(5):393–401.

Lanitis, A., Taylor, C.J., and Cootes, T.F. (1995b). A unified approach to coding and interpreting face images. In *Proc. International Conference on Computer Vision*, pages 368–373.

Leung, T.K., Burl, M.C., and Perona, P. (1995). Finding faces in cluttered secenes using random labeled graph matching. In *International Conference on Computer Vision*, pages 637–644.

Lew, M.S. (1996). Informatic theoretic view-based and modular face detection. In *International Conference on Automatic Face and Gesture Recognition*, pages 198–203.

Li, S., Zou, X., Hu, Y., Zhang, Z., Yan, S., Peng, X., Huang, L., and Zhang, H. (2001). Real-time multi-view face detection, tracking, pose estimation, alignment, and recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition*.

Lien, J. (1998). *Automatic Recognition of Facial Expressions Using Hidden Markov Models and Estimation of Expression Intensity*. PhD thesis, Carnegie Mellon University.

Littlestone, N. (1987). Learning when irrelevant attributes abound. In *Annual Symposium on Foundations of Computer Science*, pages 68–77.

Madabhushi, A. and Aggarwal, J.K. (1999). A Bayesian approach to human activity recognition. In *IEEE Workshop on Visual Surveillance Systems*, pages 25–30.

Madigan, D. and York, J. (1995). Bayesian graphical models for discrete data. *International Statistical Review*, 63:215–232.

Martinez, A. (1999). Face image retrieval using HMMs. In *IEEE Workshop on Content-based Access of Images and Video Libraries*, pages 35–39.

Mase, K. (1991). Recognition of facial expression from optical flow. *IEICE Transactions*, E74(10):3474–3483.

Matsumoto, D. (1998). Cultural influences on judgments of facial expressions of emotion. In *Proc. ATR Symposium on Face and Object Recognition*, pages 13–15.

McCallum, A.K. and Nigam, K. (1998). Employing EM in pool-based active learning for text classification. In *International Conference on Machine Learning*, pages 350–358.

McKenna, S., Gong, S., and Raja, Y. (1998). Modeling facial color and identity with Gaussian mixtures. *Patern Recognition*, 31(12):1883–1892.

McLachlan, G.J. (1992). *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley and Sons Inc., New York.

McLachlan, G.J. and Basford, K.E. (1988). *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker Inc., New York.

Mehrabian, A. (1968). Communication without words. *Psychology Today*, 2(4):53–56.

Meila, M. (1999). *Learning with Mixture of Trees*. PhD thesis, Massachusetts Institute of Technology.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller, E. (1953). Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092.

Michalski, R.S. (1983). A theory and methodology of inductive learning. *Machine Learning: an Artificial Intelligence Approach, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, eds.*, pages 83–134.

Michalski, R.S., Carbonell, J.G., and Mitchell, T.M., editors (1986). *Machine Learning: An Artificial Intelligence Approach*. Morgan Kaufmann, Los Altos, CA.

Miller, D.J. and Uyar, H.S. (1996). A mixture of experts classifier with learning based on both labelled and unlabelled data. In *Advances in Neural Information Processing Systems*, pages 571–577.

Mitchell, T. (1999). The role of unlabeled data in supervised learning. In *Proc. International Colloquium on Cognitive Science*.

MITFaceDB (2000). MIT CBCL face database, MIT center for biological and computation learning. http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html.

Moghaddam, B. and Pentland, A. (1997). Probabilistic visual learning for object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710.

Morishima, S. (1995). Emotion model: A criterion for recognition, synthesis and compression of face and emotion. In *Proc. Automatic Face and Gesture Recognition*, pages 284–289.

Murphy, K. and Paskin, M. (2001). Linear time inference in hierarchical HMMs. *Neural Information Processing Systems*.

Murphy, P.M. (1994). UCI repository of machine learning databases. Technical report, University of California, Irvine.

Nagy, G., Seth, S., and Stoddard, S. (1992). A prototype document image analysis system for technical journals. *IEEE Computer*, 25(7):10–22.

Nakamura, Y. and Kanade, T. (1997). Semantic analysis for video contents extraction - Spotting by association in news video. In *Proc. ACM International Multimedia Conference*.

Naphade, M.R. and Huang, T.S. (2000a). Semantic video indexing using a probabilistic framework. In *International Conference on Pattern Recognition*, volume 3, pages 83–89.

Naphade, M.R. and Huang, T.S. (2000b). Stochastic modeling of soundtrack for efficient segmentation and indexing of video. In *SPIE IS & T Storage and Retrieval for Multimedia Databases*, volume 3972, pages 168–176.

Naphade, M.R., Kristjansson, T., Frey, B., and Huang, T.S. (1998). Probabilistic multimedia objects (multijects): A novel approach to indexing and retrieval in multimedia systems. In *International Conference on Image Processing*, volume 3, pages 536–540.

Naphade, M.R., Wang, R., and Huang, T.S. (2001). Multimodal pattern matching for audio-visual query and retrieval. In *SPIE IS & T Storage and Retrieval for Multimedia Databases*.

Nefian, A. and Hayes, M. (1999a). An embedded HMM based approach for face detection and recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 3553–3556.

Nefian, A. and Hayes, M. (1999b). Face recognition using an embedded HMM. In *IEEE Conference on Audio and Video-based Biometric Person Authentication*, pages 19–24.

Nigam, K., McCallum, A., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39:103–134.

Nigam, K.P. (2001). Using unlabeled data to improve text classification. Technical Report CMU-CS-01-126, School of Computer Science, Carnegie Mellon University.

Oliver, N., Pentland, A., and Bérard, F. (2000). LAFTER: A real-time face and lips tracker with facial expression recognition. *Pattern Recognition*, 33:1369–1382.

O'Neill, T.J. (1978). Normal discrimination with unclassified obseravations. *Journal of the American Statistical Association*, 73(364):821–826.

Osuna, E., Freund, R., and Girosi, F. (1997). Training support vector machines: An application to face detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 130–136.

Otsuka, T. and Ohya, J. (1997a). Recognizing multiple persons' facial expressions using HMM based on automatic extraction of significant frames from image sequences. In *Proc. International Conference on Image Processing*, pages 546–549.

Otsuka, T. and Ohya, J. (1997b). A study of transformation of facial expressions based on expression recognition from temproal image sequences. Technical report, Institute of Electronic, Information, and Communications Engineers (IEICE).

Pal, S. and Pal, A. (2002). *Pattern Recognition from Classical to Modern Approaches*. World Scientific.

Pantic, M. and Rothkrantz, L.J.M. (2000). Automatic analysis of facial expressions: The state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1424–1445.

Pantic, M. and Rothkrantz, L.J.M. (2003). Toward an affect-sensitive multimodal human-computer interaction. *Proceedings of the IEEE*, 91(9):1370–1390.

Pavlovic, V. and Garg, A. (2001). Efficient detection of objects and attributes using boosting. In *IEEE Conference on Computer Vision and Pattern Recognition - Technical Sketches*.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California.

Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*. Cambridge University Press, Cambridge.

Pentland, A. (2000). Looking at people. *Communications of the ACM*, 43(3):35–44.

Picard, R. W. (1997). *Affective Computing*. MIT Press, Cambridge, MA.

Rabiner, L. and Huang, B. (1993). *Fundamentals of Speech Recognition*. Prentice-Hall.

Rabiner, L.R. (1989). A tutorial on hidden Markov models and selected applications in speech processing. *Proceedings of IEEE*, 77(2):257–286.

Rajagalopan, A., Kumar, K., Karlekar, J., Manivasakan, R., Patil, M., Desai, U., Poonacha, P., and Chaudhuri, S. (1998). Finding faces in photographs. In *International Conference on Computer Vision*, pages 640–645.

Ramesh, P. and Wilpon, J. (1992). Modeling state durations in hidden Markov models for automatic speech recognition. In *International Conference on Accoustics, Speech, and Signal Processing*, volume 1, pages 381–384.

Ratsaby, J. and Venkatesh, S.S. (1995). Learning from a mixture of labeled and unlabeled examples with parametric side information. In *Conference on Computational Learning Theory*, pages 412–417.

Redner, R.A. and Walker, H.F. (1984). Mixture densities, maximum likelihood, and the EM algorithm. *SIAM Review*, 26(2):195–239.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:368–407.

Rosenblum, M., Yacoob, Y., and Davis, L.S. (1996). Human expression recognition from motion using a radial basis function network architecture. *IEEE Transactions on Neural Network*, 7(5):1121–1138.

Roth, D. (1998). Learning to resolve natural language ambiguities: A unified approach. In *National Conference on Artifical Intelligence*, pages 806–813.

Roth, D. (1999). Learning in natural language. In *International Joint Conference of Artificial Intelligence*, pages 898–904.

Roth, D. and Zelenko, D. (2000). Towards a theory of coherent concepts. In *National Conference on Artificial Intelligence*, pages 639–644.

Rowley, H., Baluja, S., and Kanade, T. (1998a). Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38.

Rowley, H., Baluja, S., and Kanade, T. (1998b). Rotation invariant neural network-based face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 38–44.

Salovey, P. and Mayer, J.D. (1990). Emotional intelligence. *Imagination, Cognition, and Personality*, 9(3):185–211.

Samal, A. and Iyengar, P.A. (1992). Automatic recognition and analysis of human faces and facial expressions: A survey. *Pattern Recognition*, 25(1):65–77.

Schapire, R.E., Freund, Y., Bartlett, P., and Lee, W.S. (1997). Boosting the margin: A new explanation for the effectiveness of voting methods. *Machine Learning*, 29:322–330.

Schapire, R.E and Singer, Y. (1999). Improved boosting algorithms using cofidence rated predictions. *Machine Learning*, 37(3):297–336.

Schlosberg, H. (1954). Three dimensions of emotion. *Psychological Review*, 61:81–88.

Schneiderman, H. and Kanade, T. (1998). Probabilistic modeling of local appearance and spatial relationships for object recognition. In *IEEE Conference Computer Vision and Pattern Recognition*, pages 45–51.

Schneiderman, H. and Kanade, T. (2000). A statistical method for 3D object detection applied to faces and cars. In *IEEE Conference Computer Vision and Pattern Recognition*, volume 1, pages 746–751.

Sebe, N., Cohen, I., Garg, A., Lew, M.S., and Huang, T.S. (2002). Emotion recognition using a Cauchy naive Bayes classifier. In *International Conference on Pattern Recognition*, volume 1, pages 17–20.

Sebe, N. and Lew, M.S. (2003). *Robust Computer Vision – Theory and Applications*. Kluwer Academic Publishers.

Sebe, N., Lew, M.S., Cohen, I., Sun, Y., Gevers, T., and Huang, T.S. (2004). Authentic facial expression analysis. In *Automatic Face and Gesture Recognition*, pages 517–522.

Seeger, M. (2001). Learning with labeled and unlabeled data. Technical report, Edinburgh University.

Segre, A.M. (1992). Applications of machine learning. *IEEE Expert*, 7(3):31–34.

Shahshahani, B. and Landgrebe, D. (1994a). Effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *IEEE Transactions on Geoscience and Remote Sensing*, 32(5):1087–1095.

Shahshahani, B.M. and Landgrebe, D.A. (1994b). Classification of multi-spectral data by joint supervised-unsupervised learning. Technical Report TR-EE 94-1, School of Electrical Engineering, Purdue University.

Shawe-Taylor, J. (1998). Classification accuracy based on observed margin. *Algorithmica*, 22:157–172.

Shawe-Taylor, J. and Christianini, N. (2000). *An Introduction to Support Vector Machines and Other Kernel Based Methods*. Cambridge University Press.

Shawe-Taylor, J. and Cristianini, N. (1999). Further results on the margin distribution. In *Conference on Computational Learning Theory*, pages 278–285.

Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, Prediction, and Search*. MIT Press, Cambridge, 2nd edition.

Starks, H. and Woods, J. (1994). *Probability, Random Processes, and Estimation Theory for Engineers*. Prentice Hall.

Starner, T., Weaver, J., and Pentland, A. (1998). Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375.

Sung, K-K. and Poggio, T. (1998). Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51.

Tao, H. and Huang, T.S. (1998). Connected vibrations: A modal analysis approach to non-rigid motion tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 735–740.

Tefas, A., Kotroppulos, C., and Pitas, I. (1998). Variants of dynamic link architecture based on mathematical morphology for front face authentication. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 814–819.

Tishby, N., Pereira, F.C., and Bialek, W. (1999). The information bottleneck method. In *Annual Allerton Conference on Communication, Control, and Computing*, pages 368–377.

Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86.

Ueki, N., Morishima, S., Yamada, H., and Harashima, H. (1994). Expression analysis/synthesis system based on emotion space constructed by multilayered neural network. *Systems and Computers in Japan*, 25(13):95–103.

Valiant, L.G. (1984). A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.

van Allen, T. and Greiner, R. (2000). A model selection criteria for learning belief nets: An empirical comparison. In *International Conference on Machine Learning*, pages 1047–1054.

Vapnik, V.N. (1982). *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York.

Vapnik, V.N. (1998). *Statistical Learning Theory*. John Wiley and Sons, New York.

Viola, P. and Jones, M. (2004). Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154.

Wang, R.R., Huang, T.S., and Zhong, J. (2002). Generative and discriminative face modeling for detection. In *International Conference on Automatic Face and Gesture Recognition*.

White, H. (1982). Maximum likelihood estimation of misspecified models. *Econometrica*, 50(1):1–25.

Wilson, A.D. and Bobick, A.F. (1998). Recognition and interpretation of parametric gesture. In *International Conference on Computer Vision*, pages 329–336.

Wolpert, D.H. (1992). Stacked generalization. *Neural Networks*, 5:241–259.

Yacoob, Y. and Davis, L.S. (1996). Recognizing human facial expressions from long image sequences using optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):636–642.

Yang, G. and Huang, T.S. (1994). Human face detection in complex background. *Patern Recognition*, 27(1):53–63.

Yang, J. and Waibel, A. (1996). A real-time face tracker. In *Proc. Workshop on Applications of Computer Vision*, pages 142–147.

Yang, M-H., Kriegman, D., and Ahuja, N. (2002). Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine intelligence*, 24(1):34–58.

Yang, M-H., Roth, D., and Ahuja, N. (2000). SNoW based face detector. In *Neural Information and Processing Systems*, pages 855–861.

Yow, K.C. and Cipolla, R. (1997). Feature-based human face detection. *Image and Vision Computing*, 15(9):317–322.

Zacks, J. and Tversky, B. (2001). Event structure in perception and cognition. *Psychological Bulletin*, 127(1):3–21.

Zhang, T. and Oles, F. (2000). A probability analysis on the value of unlabeled data for classification problems. In *International Conference on Machine Learning*.

Index