PROBABILISTIC LEARNING: SPARSITY AND NON-DECOMPOSABLE LOSSES

Ye Nan

B.Comp. (CS) (Hons.) & B.Sc. (Applied Math) (Hons.)

NUS

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2013

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Ye Nan

31 May 2013

Acknowledgement

I would like to thank my advisors Prof. Wee Sun Lee and Prof. Sanjay Jain for their advices and encouragement during my PhD study. My experience of working with Sanjay in inductive inference has influenced how I view learning in general and approach machine learning in particular. Discussions with Wee Sun during meetings, lunches and over emails have been stimulating, and have become the source of many ideas in this thesis. I am particularly grateful to both Wee Sun and Sanjay for giving me much freedom to try out what I like to do. I would also like to thank them for reading draft versions of the thesis, and giving many comments which have significantly improved the thesis.

Besides Wee Sun and Sanjay, I would also like to thank the following.

Kian Ming Adam Chai and Hai Leong Chieu for many interesting discussions. In particular, I benefited from discussions with Hai Leong when writing my high-order CRF code, and studying Adam's work and discussing with him on optimizing F-measures.

Assistant Prof. Bryan Kian Hsiang Low, A/P Tze Yun Leong, and Stephen Gould for many helpful comments which help improving the presentation and quality of the thesis significantly.

Prof. Frank Stephan and A/P Hon Wai Leong, for valuable research experience that I had when working with them.

Last but not least, I would like to thank my family for their love and support.

Contents

Abstract

Machine learning is concerned with automating information discovery from data for making predictions and decisions, with statistical learning as one major paradigm. This thesis considers statistical learning with structured data and general loss functions.

For learning with structured data, we consider conditional random fields (CRFs). CRFs form a rich class of structured conditional models which yield state-of-the-art performance in many applications, but inference and learning for CRFs with general structures are intractable. In practice usually only simple dependencies are considered or approximation methods are adopted. We demonstrate that sparse potential functions may be an avenue to exploit for designing efficient inference and learning algorithms for general CRFs. We identify two useful types of CRFs with sparse potential functions, and give efficient (polynomial time) exact inference and learning algorithms for them. One is a class of high-order CRFs with a particular type of sparse high-order potential functions, and the other is a class of factorial CRFs with sparse co-temporal potential functions. We demonstrate that these CRFs perform well on synthetic and real datasets. In addition, we give algorithms for handling CRFs incorporating both sparse high-order features and sparse co-temporal features.

For learning with general loss functions, we consider the theory and algorithms of learning to optimize F-measures. F-measures form a class of non-decomposable losses popular in tasks including information retrieval, information extraction and multi-label classification, but the theory and algorithms are still not yet quite well understood due to its non-decomposability. We first give theoretical justifications and connections between two learning paradigms: the empirical utility maximization (EUM) approach learns a classifier having optimal performance on training data, while the decision-theoretic approach (DTA) learns a probabilistic model and then predicts labels with maximum expected F-measure. Given accurate models, theory suggests that the two approaches are asymptotically equivalent given large training and test sets. Empirically, the EUM approach appears to be more robust against model misspecification, whereas given a good model, the decision-theoretic approach appears to be better for handling rare classes and a common domain adaptation scenario. In addition, while previous algorithms for computing the expected F-measure require at least cubic time, we give a quadratic time algorithm, making DTA a more practical approach.

List of Tables

List of Figures

Notations

The following are notational conventions followed throughout the thesis, unless otherwise stated. They mostly follow standard notations in the literature, and thus may be consulted only when needed.

Abbrevations

Various notational details are often omitted to ease reading, as long as such omission does not create ambiguity.

For example, in a summation or integration notation, the range of summation or integration is often omitted if it is clear from the context. In particular, in notations like $\sum_x$ or $\int f(x)dx$, if the range of x is not explicitly mentioned, then it is assumed to be the universe of discourse for x.

Probability

A *random variable* is generally denoted by a capital letter such as X, Y, Z, while their domains are often denoted by $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$ and so on. An *instantiation* of the random variable is denoted by the corresponding lower case letter.

$P(X)$ represents a probability distribution on the random variable X. It is the *probability mass function* (pmf) if X is discrete, and is the *probability density function* (pdf) if X is continuous. $P(x)$ denotes the value of $P(X)$ when X is instantiated as x. $\mathrm{E}_{X\sim P}(X)$ denotes the *expectation* of a random variable X following distribution P, which is often abbreviated as $\mathrm{E}(X)$ if P is clear from the context. A notation like $\mathrm{E}_{X_1}(f(X_1, X_2))$ indicates taking expectation with respect to X_1 only.

$P(Y|X)$ represents a conditional probability distribution of Y given X, which is either a pmf or a pdf depending on whether Y is discrete or continuous.

Given a joint distribution $P(X_1, \ldots, X_n)$ for random variables $X_1, \ldots, X_n$, and a subset S of $\{X_1, \ldots, X_n\}$, P_S is used to denote the *marginal distribution* derived from

$P(X_1, \ldots, X_n)$ for random variables in S. If $S_1, S_2 \subseteq \{X_1, \ldots, X_n\}$, then $P_{S_1|S_2}$ denote the conditional distribution derived from $P(X_1, \ldots, X_n)$ for random variables in S_1 given S_2. The subscripts are often omitted when it is clear from the contexts which random variables are considered. For example, given $P(X_1, X_2)$, the conditional distribution $P_{X_1}(x_1) \stackrel{\text{def}}{=} \sum_{x_2} P(x_1, x_2)$ is often written as $P(X_1)$. Similarly, $P_{X_1|X_2}(x_1|x_2) = \frac{P(x_1,x_2)}{P(x_2)}$ is often written as $P(X_1|X_2)$.

Linear algebra

A matrix is generally denoted by capital letters like A, B, C. The entries in the matrix is often named using the corresponding lower case letter indexed with its row and column numbers as the subscript. For example, if A is a matrix, then we often use a_{ij} to denote the entry in the i-th row and j-th column. Alternatively, an $n \times m$ matrix where the (i, j)-th entry is a_{ij} can be written as $(a_{ij})_{n,m}$, or simply (a_{ij}) if n, m are clear from context. The determinant, inverse and transpose of a matrix A are denoted by $|A|$, A^{-1} and A^T respectively. The (i, j)-minor of A, that is, the matrix obtained by deleting the i-th row and the j-th column of A, is generally denoted by A_{ij}.

A vector is a column vector unless otherwise stated, and is often denoted by boldfaced lower case letters.

Some common notations

$\mathbf{R}$	the set of real numbers
$\|\|\mathbf{x}\|\|_p$	the ℓ_p norm of the vector $\mathbf{x}$
$\text{diag}(a_1, \ldots, a_n)$	$n \times n$ diagonal matrix with a_i's as the diagonal entries
$\text{I}(\cdot)$	the indicator function which is 1 if $\cdot$ is true, and 0 otherwise
$N(\mu, \Sigma)$	normal distribution with mean μ and covariance matrix Σ
$N(\mathbf{x}; \mu, \Sigma)$	$(2\pi)^{-k/2}\|\Sigma\|^{-1/2} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)}$, the pdf for a k-dimensional normal distribution $N(\mu, \Sigma)$
$\boldsymbol{\nabla} f$	gradient of f; subscript may be added to indicate the gradient with respect to a subset of variables with others fixed

Chapter 1

Introduction

Statistical methods have become a franca lingua in machine learning. In classifical statistical learning, sound theoretical principles (Vapnik, 1998) and effective algorithms (Hastie et al., 2005) have been developed for problems like classification, regression and density estimation. In recent years, the widespread use of machine learning as an enabling technology for automating information discovery for decision making and prediction has led researchers and practitioners to work on increasingly more complex data with complex performance measures. Various interesting problems and challenges have emerged. This thesis is motivated by the goal of moving towards a more general statistical framework for dealing with complex data and performance measures. Its contribution consists of identifying subclasses within the useful but useful generally intractable family of conditional probability distributions called Conditional Random Fields (CRFs). Efficient exact inference and learning algorithms are developed for handling these dependencies. In addition, theoretical and empirical analysis and comparisons are done on algorithms for maximizing a popular class of performance measures, F-measures, which are non-decomposable and poses different theoretical and algorithmic challenges as compared to performance measures like accuracy.

For learning with structured data, a traditional approach is to reduce the problems

to classification problems. Such problems include parts of speech tagging (Brill, 1994), coreference resolution (Soon et al., 2001) and relation extraction (Zelenko et al., 2003). Such a reduction generally loses useful dependencies between the instances, and do not perform as well as models incorporating structural dependencies. However, modeling structures generally lead to difficult computational problems. In particular, learning and inference are generally intractable (Istrail, 2000) for an important class of structured statistical models, Markov random fields (MRFs) (Kindermann and Snell, 1980) and its conditional version, conditional random fields (CRFs) (Lafferty et al., 2001). Two approaches are thus often used in practice: include only simple local dependencies, or apply efficient approximation methods.

Incorporating only simple local dependencies can risk significant loss of information. For example, consider linear-chain CRFs Lafferty et al. (2001), a popular model using only simple pairwise dependencies, and satisfying the Markov property that knowledge of the previous label is irrelevant to the next label once the current label is known. This makes computationally efficient algorithms feasible. In many cases, linear-chain CRFs serve as a reasonable approximation to reality, for example, when inferring the parts of speech of a sentence. However, in other applications, performance can be improved if higher order dependencies are considered. For example, in inferring handwritten characters within words, knowledge of the previous few characters can provide a lot of information about the identity of the next character. Such dependencies can be captured using high-order CRFs, an extension of linear-chain CRFs to capture the higher order dependencies. However, the time complexity of typical inference and learning algorithms for high-order CRFs are exponential in the order, and quickly becomes infeasible.

Another example requiring modeling beyond simple pairwise dependencies is sequence multi-labeling, which involves labeling an observation sequence with multiple dependent sequences. For example, in activity recognition, we may be interested in

labeling a sequence of sensor inputs with whether a person is exercising at each time instance, and with whether the person is listening to music at each time instance. In this case, there are dependencies not only between consecutive time instances, but also across the two different activities as people often exercise and listen to music at the same time. These dependencies can be captured using factorial CRFs (FCRFs) (Sutton et al., 2007). However, with increasing number of chains, inference and learning for FCRFs become computationally intractable without additional assumptions.

While approximation algorithms serve as practical means to deal with complex dependencies given limited computation time, their behavior can be hard to predict. For example, loopy belief propogation is often used as an approximate inference method for graphical models, and have been shown to work well for various problems, but they can produce results which oscillates and are not truly related to the correct ones (Murphy et al., 1999).

In this thesis, we demonstrate that expressiveness and exactness can be achieved at the same time in some cases. The key observation is that certain useful complex dependencies are sparse. For example, in handwritten character recognition, the number of character patterns is very small as compared to all character combinations. In the case of sequence multi-labeling, the number of co-temporal patterns may also be relatively small. Another interesting example of sparse complex dependency is the set of co-temporal patterns predicted by classifiers in sequence multi-labeling. For such sparse models, the sufficient statistics required in inference and learning can be represented compactly and evaluated efficiently. In our case, we identify a class of sparse high-order CRFs and a class of sparse FCRFs for which we design exact polynomial time inference and learning algorithms. While the techniques used are different for sparse high-order CRFs and sparse FCRFs, we give an algorithm to handle CRFs with sparse higher order features in the chains and sparse co-temporal features. The time complexity of the algorithm can grow exponentially in the number of chains, as in the

case of naive generalizations of linear-chain algorithms for FCRFs, but can be efficient if the number of chains using high-order features is small.

Besides the interests in developing better means for modeling data, there has also been an increasing interests in and extensive use of general loss functions, other than traditional performance measures like accuracy and square loss. For example, when the dataset is imbalanced (i.e. some classes are rare in comparison to other classes), then predicting the most common classes will often result in high accuracy. In this case, one class of commonly used utility functions are the F-measures, which measure the performance of a classifier in terms of its ability to obtain both high recall (recover most of the instances in the rare classes) and high precision (instances predicted to be in the rare classes are mostly truly rare). A main difference between accuracy and F-measures is that while accuracy can be expressed as a sum of the contribution from the instances, F-measures cannot. We call accuracy as a decomposable utility function and F-measure a non-decomposable utility function. The study of the theory and algorithms of F-measures are attractive due to their increasing popularity in information retrieval (Manning et al., 2009) information extraction (Tjong Kim Sang and De Meulder, 2003), and multi-label classification (Dembczynski et al., 2011). Another type of commonly used non-decomposable utility function is the AUC (Area under the ROC Curve) score (Fawcett, 2006). However, non-decomposability poses new theoretical and algorithmic challenges in learning and inference, as compared to those for decomposable losses.

In this thesis, we study only the simplest setting for non-decomposable utility/loss function, that of labeling binary independent identically distributed examples. We also focus mainly on the F-measure. We give theoretical justifications and connections between different types learning algorithms. We also give efficient algorithms for computing optimal predictions, and carry out empirical studies to investigate the performance of different algorithms.

1.1 Contributions

We first demonstrate that under realistic sparsity assumptions on features, it is possible to design efficient exact learning and inference algorithms to handle dependencies beyond pairwise dependencies in CRFs. We consider sparse high-order features (features depending on several consecutive labels) for labeling observations with a single label sequence, and sparse co-temporal features for labeling observation with multiple label sequences. Our inference and learning algorithms are exact and have polynomial time complexity. Both types of features are demonstrated to yield significant performance gains on some synthetic and real datasets. The techniques used for exploiting sparsity in high-order CRFs and FCRFs are different, and we discuss an algorithm combining these two techniques to perform inference and learning for CRFs with sparse high order features in the chains and sparse co-temporal features. The main insight in our results is that natural sparse features form an avenue that can be exploited to yield efficient algorithms. In our case, we exploit sparsity by modifying existing algorithms to derive compact representations for quantities of interests.

We then consider learning with non-decomposable utility functions, focusing on F-measures. We first demonstrate that F-measures and several other utility functions are non-decomposable, thus the classical theory for decomposable utility functions no longer applies. We then give theoretical justifications and connections between two learning paradigms for F-measures: the empirical utility maximization (EUM) approach learns a classifier having optimal performance on training data, while the decision-theoretic approach (DTA) learns a probabilistic model and then predicts labels with maximum expected F-measure. For the EUM approach, we show that it learns an optimal classifier in the limit, and we justify that a simple thresholding method is optimal. For the DTA approach, we give an $O(n^2)$ time algorithm for computing the predictions with maximum expected F-measure. Given accurate models, theory suggests that the two approaches are asymptotically equivalent given large training and

test sets. Empirically, the EUM approach appears to be more robust against model misspecification, and given a good model, the decision-theoretic approach appears to be better for handling rare classes and a common domain adaptation scenario.

There are also various small contributions in the background material on statistical learning and the exponential families, which are motivated by questions that can make the discussion more self-contained, but of which I was not aware of any published answer. Whenever possible, I gave my own solutions – though sometimes only high-level ideas were given. Example questions include how the quality of an estimated distribution affect its prediction performance, how irrelevant attributes affect the final hypothesis learned and the convergence rate, what is the quality of the linear classifier learned by using the square loss or the exponential loss as the surrogate losses for the 0/1 loss, whether the well-known consistency result for maximum likelihood estimation of generative distributions holds for conditional distributions. Examples or technical derivations are also given whenever possible. Certain technical derivations are simplified, generalized or with new results given. In particular, simplified proofs are given for Wolpert's no free lunch theorem and Hammersley-Clifford theorem. A discussion is given on an alternative definition of entropy. A derivation on conditional exponential family as maximum entropy models is given under general equality and inequality constraints. There is also a result on the independence property of Markov random fields.

1.2 Outline

Chapter 2 is on the statistical framework of decision and learning, and Chapter 3 is on log-linearity and Markov property in undirected graphical models. They lay the foundation to the main contribution of this thesis, namely, the sparse high-order CRF in Chapter 4, the sparse FCRF in Chapter 5, and the theory and algorithms

for F-measures in Chapter 6. I hope the first two chapters will also make the thesis as self-contained as possible. While most things in these two chapters are old, they have been organized and presented differently, with some new results as mentioned in previous section. They also reflect questions which I asked when I started studying machine learning, but did not manage to find answers easily. The following is a more detailed outline of each chapter.

Chapter 2 presents the statistical framework of decision and learning. The insights from the results presented in this chapter shaped many aspects of the thesis, which will be too many to enumerate. For example, the principle of using regularization has been elaborated in detail, and used freely in later chapters as a standard practice to achieve stability, to incorporate prior domain knowledge, or to control the tradeoff between quality of fit to data and the complexity of hypothesis. The No-Free-Lunch theorems motivated the exploration of what algorithms work better under specific assumptions in the theoretical analysis of F-measures. This chapter first starts with the principles of decision and learning in a statistical setting. The principles are demonstrated to provide language and tools for systematic interpretation, design, and analysis of machine learning algorithms. The design of learning machines is then decomposed as representation, approximation, learning and prediction, with each component analyzed based on the basic principles. The importance of prior knowledge in machine learning is discussed.

The discussion focuses on answering two questions: (a) Are there unifying principles behind the particularities of distinct learning algorithms? (b) What are the limitations of machine learning? While most results presented are well-known, I hope the presentation shows a unified and systematic framework towards the design and analysis of machine learning algorithms, and highlights some generalization difficulties in machine learning.

Chapter 3 presents the theory of log-linear models (or exponential families) and

MRFs. Log-linearity and Markov property combine together to yield parametric models for which efficient inference and learning are possible, and our sparse models fall within the framework of log-linear CRFs. The focus is on the principles leading to such models, and the principles of inference and learning. This includes the derivation of the exponential families as maximum entropy models, the derivation of MRFs and CRFs as a consequence of a Markov property on graphs, computational difficulties for inference with exponential families, and maximum likelihood parameter estimation.

Chapter 4 presents new efficient (polynomial time) exact inference and learning algorithms for exploiting a type of sparse features in high-order CRFs for sequence labeling and segmentation. We discuss the effect of omitting inactive features and provide a justification for using only seen label patterns in features. Sparse high-order CRFs are shown to perform well on synthetic and real datasets. Conditions favoring the sparse features are discussed.

Chapter 5 presents new efficient (polynomial time) exact inference and learning algorithms for exploiting a type of sparse co-temporal features in factorial CRFs for jointly labeling and segmentation of sequences. Sparse factorial CRFs are shown to perform well on synthetic and real datasets. We also discuss an inference algorithm for CRFs with both sparse co-temporal features and sparse high-order features.

Chapter 6 presents results on learning with non-decomposable utility functions, focusing on F-measures. We first demonstrate that F-measures and several other utility functions are non-decomposable, thus the classifical theory for decomposable utility functions no longer apply. Theoretical justifications and connections for the EUM approach and the DTA approach for learning to maximize F-measures. Given accurate models, theoretical analysis suggest that the two approaches are asymptotically equivalent given large training and test sets. Empirically, the EUM approach appears to be more robust against model misspecification, and given a good model, the decision-theoretic approach appears to be better for handling rare classes and a common domain

adaptation scenario.

Chapter 7 concludes the thesis by summarizing the contributions, and highlighting problems for which the solutions can lead to deeper understanding on exploiting structural sparsity and learning to optimize non-decomposable utility/losses.

Chapter 2

Statistical Learning

Machine learning is concerned with automating information discovery from data for making predictions and decisions. Since the construction of the Perceptron (Rosenblatt, 1962) as the first learning machine in the 1960s, many learning algorithms have been proposed, including decision trees, maximum entropy classifiers, support vector machines, boosting, graphical models, for various problems. Two questions are fundamental in understanding and designing machine learning algorithms.

First, are there unifying principles behind the particularities of apparently quite distinct algorithms such as decision trees, logistic regression, support vector machines and artificial neural networks? This is a question one usually ask when confronted with the particularities of different machine learning algorithms. For example, why is the particular form logistic regression used, other than that linear functions are simple? Is naive Bayes purely based on the frequentist perspective of estimating marginals by frequencies? Why should parameters of logistic regression be estimated by maximizing likelihood although no choice of parameters can yield a model identical to the true one?

Second, what are the limitations of machine learning? A truly general-purpose learning machine is one equipped with some basic functionalities like speech or image

processing capabilities, and learns to perform new tasks such as playing chess, solving calculus problems, helping programmers to debug. However, human learning is still far from being perfectly understood, and the sense of learning in machine learning was and is still very domain specific – while it is easy for a child to learn to play new board games like chess, a computer can learn to play chess only if there is a program that is built for this particular task. Although it is still not very well understood what machines cannot learn, a sense of what machines need to learn is essential.

This chapter presents fundamental ideas in machine learning, mainly from a statistical perspective, and directed to answer the above two problems. Most results presented are well-known, my aim is to present them in the most general form or in a simplified form, and provide comprehensive discussions on some problems. There are also some new examples and new results. I hope the presentation shows a unified and systematic framework towards the design and analysis of machine learning algorithms, and highlights some generalization difficulties in machine learning. Certainly, every model has its own limitations, and the framework of statistical learning makes assumptions which may not be satisfied by the data which one works with. But the general principle is that with suitable assumptions on data and a precise performance measure, in principle one can derive conclusions about an algorithm's performance.

Here is an outline of this chapter. Section 2.1 provides a very brief overview of machine learning and discusses the role of data generation mechanisms and performance measures in the design of machine learning algorithms. Section 2.2 presents the assumptions on data generation mechanisms and the performance measures used in statistical decision and learning. Basic principles for statistical decision and learning are described and illustrated with several classical learning algorithms. Section 2.3 presents the design of a learning system as solving four subproblems: representation, approximation, estimation and prediction. Difficulties and techniques for each subproblem are discussed, mostly based on the statistical framework set up in Section 2.2.

Section 2.4 discusses theoretical results on the necessity of prior knowledge for designing learning algorithms that are guaranteed to learn the true laws. Section 2.5 discusses problems that remain to be solved.

2.1 Introduction

2.1.1 Overview

Since the ancient times, the idea of artificial intelligence captured and continue to capture the imagination of men. For example, Homer described in the *Iliad* the *Golden Servants*, which were intelligent and vocal automatons created by Hephaestus out of metal. Leibniz attempted to construct a universal representation of ideas, known as the alphabet of human thought, which can reduce much reasoning to calculations. Wolfgang von Kempelen built and showcased a sensational fake chess playing machine called the Turk. Mary Shelley described in *Frankenstein* a human created intelligent monster. And modern science fiction writers imagined intelligent robots.

It is not yet fully understood what makes humans intelligent, but learning seems to be essential for acquiring abilities to perform intelligent activities, ranging from speech to games and scientific discovery. Learning is also adopted as the solution to build reliable systems working in uncertain and noisy environments, such as household assistant robots, dialog-based expert systems, and automatic component design in manufacturing industry. It is infeasible to hardcode the behavior of these systems for all possible circumstances.

But the first learning machine, the Perceptron, was only constructed in the early 1960s by Rosenblatt. Its design simulated human neural network, and was used for the task of recognizing handwritten characters (Rosenblatt, 1962). Many successful learning systems have been built since then, for tasks such as automatic driving, playing board games, natural language processing, spam detection, credit card fraudulence

analysis. A number of deployed systems are described in (Langley and Simon, 1995). These successes are empowered by the discovery of general learning methods, such as artificial neural networks (Anthony and Bartlett, 1999; Haykin, 1999), rule induction (Quinlan, 1993), genetic algorithms (Goldberg, 1994), case-based learning (Aamodt and Plaza, 1994), explanation-based learning (Ellman, 1989), statistical learning (Vapnik, 1998), and meta-learning algorithms such as bagging (Breiman, 1996) and boosting (Freund and Schapire, 1997). At the same time, theoretical developments have yielded insightful interpretations, design techniques, and understanding of the properties, connections and limitations of learning algorithms. Notable theoretical models of learning include statistical learning (Vapnik and Chervonenkis, 1971), Valiant's PAC-learning (Valiant, 1984), and the inductive inference model studied by Solomonoff (Solomonoff, 1964) and Gold (Gold, 1967).

Machine learning is now used to help understanding the nature and mechanism of human learning, and as a tool for constructing adaptive systems which automatically improves performance with more experience. Machine learning is a cross-discipline field, drawing motivation from cognitive science, deriving its problems from areas like natural language processing, robotics, computer vision, and uses as its tools for modeling and analysis disciplines like logic, statistical science, information theory, and complexity theory.

However, machine learning is still a young field. Despite significant progresses towards the understanding and automation of learning, there are still many fundamental problems that need to be addressed, and the construction of an effective learning system is still highly nontrivial and often a very laborious task.

2.1.2 The Concept of Machine Learning

Machine learning is any algorithmic process which yields improved performance as the amount of available data grows. While the problems solved by machine learning range

from pattern recognition, regression to clustering, and many different algorithms have been designed for solving these problems, the design of an effective learning algorithm requires consideration over two key factors: the nature of data generation mechanisms and the performance measures used. This is particularly so when one engineers a general algorithm to work for a particular problem.

A data generation mechanism generates observations of a collection of variables. Learning problems are often categorized by the nature of the data generation mechanisms, and learning algorithms of very different nature are designed in each case. For example, based on whether the labels are always observed in training data, labeling problems (such as labeling whether an email is a spam, or labeling the parts of speech tags for a sentence) can be categorized as *supervised learning* (all labels observed) (Kotsiantis et al., 2007), *semi-supervised learning* (some labels observed) (Zhu, 2005), and *unsupervised learning* (no label observed) (Ghahramani, 2004). Based on the relationship between the generation mechanisms for training and test data, learning can be classified as *single-task learning* (identical mechanisms), *domain-adaptation* (same set of variables but different mechanisms), and *transfer learning* (different sets of variables for the generation mechanisms).

The simplest and most commonly used performance measures for learning algorithms are empirical measures defined on data samples, such as accuracies. However, in general, if algorithm A has better empirical performance than algorithm B on a test sample, it does not mean A always performs better than B on other samples, in particular, on the unseen examples; and in the worst case, B may perform better than A. Nevertheless, for certain cases, empirical measures do reveal useful information about learning algorithms. For example, for classification on i.i.d. instances, if A has better accuracy than B on a very large test set, then it is very likely that A has higher accuracy than B on another large test set. In this case, one may compare algorithms based on their expected accuracies, for which small confidence intervals can often be

inferred with high confidence if there are sufficiently many test examples. In any case, when comparing the performance of two machine learning algorithms, one should be aware of the information revealed by the empirical measures, and bear in mind there may be subtle considerations as above.

2.2 Statistical Decision and Learning

Statistical methods have become a franca lingua in machine learning due to their theoretical generality and empirical successes. They are based on probabilistic assumptions on the data generation mechanisms. We first present the principles of statistical decision and learning, then use some classic learning algorithms to illustrate how they provide a general framework for interpreting and analyzing learning algorithms. The example algorithms discussed include one regression algorithm (linear least squares regression), two classification algorithms (naive Bayes and nearest neighbor), and one domain adaptation algorithm (instance weighting). For each algorithm, we describe the method, show how statistical learning and decision theory can be applied to interpret it. Performance guanrantees and alternatives are discussed whenever possible.

2.2.1 Principles

A.*Statistical decision.* Statistical decision theory is concerned with making optimal predictions on data generated according to a given distribution. While the objective of a decision problem can be maximizing certain utility or minimizing certain risk, we focus on risk minimization here. Similar principles and results can be stated for utility maximization.

The general principle of statistical decision is as follows. Let P be a distribution on $(\mathcal{X} \times \mathcal{Y})^*$ [1], where $\mathcal{X}$ is the set of possible observations (or inputs), and $\mathcal{Y}$ is the

[1]That is, P is a distribution on sequences of elements from $\mathcal{X} \times \mathcal{Y}$

set of possible outcomes (or outputs, or labels). Intuitively, given test observations $x_1, \ldots, x_n \in \mathcal{X}$, the goal is to predict $s_1, \ldots, s_n \in \mathcal{Y}$ which minimize a task-specific *loss function* $L(s_1, \ldots, s_n, y_1, \ldots, y_n)$ on the average case, where $y_1, \ldots, y_n \in \mathcal{Y}$ are the true outcomes. Formally, given $x_1, \ldots, x_n$, predict

$$\arg \min_{s_1,\ldots,s_n} \mathrm{E}_{Y_1,\ldots,Y_n|x_1,\ldots,x_n \sim P}(L(s_1, \ldots, s_n, Y_1, \ldots, Y_n)), \tag{2.2.1}$$

where the expectation is over $Y_1, \ldots, Y_n$ drawn according to the conditional distribution of the true outcomes for $x_1, \ldots, x_n$ according to P.

Typically, we consider predictions of a rule $h : \mathcal{X} \to \mathcal{Y}$ on a stream of independent and identically distributed (*i.i.d.*) test points $x_1, x_2, \ldots$, using a decomposable loss function $L(s_1, \ldots, s_n, y_1, \ldots, y_n) \stackrel{\text{def}}{=} \sum_{i=1}^{n} L(s_i, y_i)$. $L(s_i, y_i)$ will be called a loss function in such setting, and we use P to denote the distribution on $\mathcal{X} \times \mathcal{Y}$ instead of $(\mathcal{X} \times \mathcal{Y})^*$. In this case, the performance metric of h can be formalized as its *expected risk*

$$\mathrm{R}(h) \stackrel{\text{def}}{=} \mathrm{E}(L(h(X), Y)) = \sum_{x,y} P(x, y) L(h(x), y) \tag{2.2.2}$$

The goal is to find a function h^* which minimizes $\mathrm{R}(h)$. h^* is called the *Bayes optimal prediction rule* and $\mathrm{R}(h^*)$ is called the *Bayes risk*.

When $\mathcal{X}$ is a metric space and $\mathcal{Y}$ is finite, it is useful to describe a prediction rule using the concept of decision boundary. For any prediction rule h, $\mathcal{X}$ is partitioned into sets consisting of elements mapping to the same y value. The union of the boundaries of such sets, if exist, is called the *decision boundary* of h. The decision boundary of h^* is called the *Bayes decision boundary*.

We illustrate the above concepts in the following example.

Example 1. Consider classifying any $x \in \mathbf{R}^2$ to a $y \in \{0, 1\}$, where the joint distri-

bution $P(X, Y)$ is as follows:

$$\begin{aligned} P(X,Y) &= \pi(Y) f(X|Y), \text{where} \\ \pi(0) &= \pi(1) = \frac{1}{2}, \\ f(X|0) &= N[(0,0), \text{diag}(1,1)], \\ f(X|1) &= N[(1,1), \text{diag}(1,1)] \end{aligned}$$

Using the *0/1 loss* $L_{01}(y_1, y_2) = \mathrm{I}(y_1 \neq y_2)$, the Bayes decision boundary is given by

$$x_1 + x_2 = 1$$

The Bayes risk can be shown to be given by

$$1 - \Phi(\sqrt{2}/2) = 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\sqrt{2}/2} e^{-x^2/2} \approx 1 - 0.7602 = 0.2398$$

In general, let X be a random n dimensional vector, Y be a random variable taking values $1, \ldots, K$. If $P(Y = k) = \frac{1}{K}$, and $P(X = x|Y = k)$ is the normal distribution $N(\mu_k, I)$, then the Bayes decision boundary is the Voronoi diagram for $\mu_1, \ldots, \mu_K$. □

B.*Statistical learning.* In statistical learning, the distribution P is now unknown but fixed. The data generation process is generally assumed to generate i.i.d. sequence $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ drawn from P. The task is to learn a good prediction rule from data.

There are three main approaches in statistical learning: risk minimization (Vapnik, 1998), density estimation, and Bayesian learning (Lindley, 1972). In the following, let $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a set of training examples, $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$.

The risk minimization approach assumes a hypothesis space $\mathcal{H}$ consisting of all

candidate prediction rules, and uses the training data to select a hypothesis which has potentially smallest risk. Formally, the *empirical risk* of a hypothesis h on D is defined as $\mathrm{R}_n(h) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} L(h(x_i), y_i)$. Note that $\mathrm{R}_n(h)$ is a random variable. A natural criterion for risk minimization is the *Empirical Risk Minimization* (*ERM*) principle, which selects the hypothesis with minimal empirical risk:

$$h_n \stackrel{\text{def}}{=} \arg\min_{h\in\mathcal{H}} \mathrm{R}_n(h) \tag{2.2.3}$$

The density estimation approach first uses the training data to compute an estimate $\tilde{P}(X,Y)$ for the joint distribution $P(X,Y)$ or an estimate $\tilde{P}(Y|X)$ for the conditional distribution $P(Y|X)$. Then prediction is made to minimize the risk with respect to the estimated distribution $\tilde{P}(Y|X)$. Given a class of candidate joint distribution $\{P(X,Y|\theta) : \theta \in \Theta\}$, where θ is an index, the maximum likelihood (ML) distribution is often used as the estimate:

$$\tilde{P}_n(X,Y) = \arg\max_{\theta\in\Theta} \prod_i P(x_i, y_i|\theta) \tag{2.2.4}$$

Similarly, for the conditional case, the maximum likelihood estimate (MLE) is

$$\tilde{P}_n(Y|X) = \arg\max_{\theta\in\Theta} \prod_i P(y_i|x_i, \theta) \tag{2.2.5}$$

In Bayesian learning, there is a prior distribution $P(\theta)$ on the set $\{P(X,Y|\theta : \theta \in \Theta\}$ of possible data distributions. Given training data D, and test examples $\mathbf{x}' = (x'_1, \ldots, x'_m)$, the posterior distribution $P(\mathbf{y}'|\mathbf{x}', D)$ for $\mathbf{y}' = (y'_1, \ldots, y'_m)$ is then used for making predictions. If joint distributions are considered, the posterior distribution is computed using the Bayes rule as follows:

$$P(\mathbf{y}'|\mathbf{x}', D) = \textstyle\int_\theta P(\mathbf{y}', \mathbf{x}', D|\theta)P(\theta)d\theta / P(\mathbf{x}, D) \tag{2.2.6}$$

Similarly, for conditional distributions, the posterior distribution is computed as follows:

$$P(\mathbf{y}'|\mathbf{x}', D) = \int_\theta P(\mathbf{y}', \mathbf{y}|\mathbf{x}', \mathbf{x}, \theta)P(\theta)d\theta / P(\mathbf{y}|\mathbf{x}) \tag{2.2.7}$$

For all the above approaches, generalization can occur only if the set of hypotheses is not too expressive. For example, consider classifying points in $\mathbf{R}^2$. If all possible classifiers are allowed, then for any classifier, there are infinitely many other classifiers which agree with it on the training data, but differ drastically from it on unseen points. The choice of the hypothesis space depends on prior knowledge about the problem. A consequence of the restricted set of hypotheses is that it may not always be possible to construct from data a sequence of hypotheses that converges to the Bayes optimal hypothesis.

The above three approaches are have their own advantages and disadvantages but precise comparisons on their performance cannot be made without further assumptions. We make a few remarks on some general connections.

First, only the risk minimization approach is directly concerned with minimizing risk, thus in principle, it is at least as good as the other two approaches in terms of the ability to minimize risks. We elaborate on this by comparing the risk minimization approach and the density estimation approach. If the hypothesized set of densities is $\mathcal{P}$, and for $P \in \mathcal{P}$, the Bayes optimal prediction rule is h_P, then we can form $\mathcal{H} = \{h_P : P \in \mathcal{P}\}$, and apply the risk minimization on $\mathcal{H}$. If P^* is the optimal density in $\mathcal{P}$ (it does not matter how optimality is defined), and h^* is a hypothesis in $\mathcal{H}$ with minimum risk, then $\mathrm{R}(h^*) \leq \mathrm{R}(h_{P^*})$. Thus in the limit, the risk minimization approach is at least as good as the density estimation approach if the risk of the estimated hypothesis from $\mathcal{H}$ eventually converge to $\mathrm{R}(h^*)$. However, the risk minimization approach is not necessarily preferred, because it can lead to difficult computational problems.

Second, the density estimation approach and Bayesian learning requires training once only even though we may be interested in several different loss functions, while

for the risk minimization approach, training need to be done for each loss function.

Third, when the training set size is large enough, the posterior distribution in Bayesian learning is often close to the maximum likelihood distribution. When the training set size is small, the relative performance of Bayesian learning and density estimation mainly depend on the quality of the prior.

2.2.2 Least Squares Linear Regression

A.*Method.* Regression is the estimation of the functional relationship between an observation variable X in $\mathbf{R}^d$ and a real output variable Y.

Suppose a set of (X, Y) pairs, $(x_1, y_1), \ldots, (x_n, y_n)$ are collected from experiments and plotted as in Fig 2.1(a). Y appears to be approximately linear in x. The least squares criterion chooses the best fitting line $Y = aX + b$ as the one minimizing the *residual sum of squares*

$$\mathrm{R}(a, b) \stackrel{\text{def}}{=} \sum_{i=1}^{n} (ax_i + b - y_i)^2. \tag{2.2.8}$$

The minimum of R occurs at (a^*, b^*) which are solutions to the equations

$$\frac{\partial \mathrm{R}}{\partial a} = \sum_{i=1}^{n} 2(ax_i + b - y_i)x_i = 0, \text{ and } \frac{\partial \mathrm{R}}{\partial b} = \sum_{i=1}^{n} 2(ax_i + b - y_i) = 0.$$

Solving the equations, we have

$$a^* = \frac{\tilde{\mathrm{E}}(XY) - \tilde{\mathrm{E}}(X)\,\tilde{\mathrm{E}}(Y)}{\tilde{\mathrm{E}}(X^2) - \tilde{\mathrm{E}}(X)^2}, \text{ and } b^* = \frac{\tilde{\mathrm{E}}(Y)\,\tilde{\mathrm{E}}(X^2) - \tilde{\mathrm{E}}(X)\,\tilde{\mathrm{E}}(XY)}{\tilde{\mathrm{E}}(X^2) - \tilde{\mathrm{E}}(X)^2} = \tilde{\mathrm{E}}(Y) - a^*\,\tilde{\mathrm{E}}(X),$$

where $\tilde{\mathrm{E}}(X) = \frac{1}{n}\sum_{i=1}^{n} x_i$, $\tilde{\mathrm{E}}(X^2) = \frac{1}{n}\sum_{i=1}^{n} x_i^2$, $\tilde{\mathrm{E}}(Y) = \frac{1}{n}\sum_{i=1}^{n} y_i$, $\tilde{\mathrm{E}}(XY) = \frac{1}{n}\sum_{i=1}^{n} x_i y_i$.

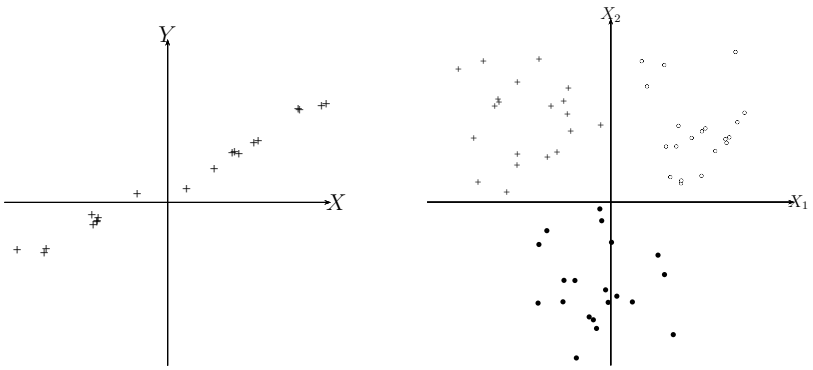
While the case for $d = 1$ seems complicated, least squares linear regression in fact has an elegant general solution. To simplify notations, let x_i denote the vector obtained by prefixing the original x_i by 1. Let $\mathbf{X}$ be the $n \times (d+1)$ matrix with x_i^T as its ith row, then a hyperplane for the original data can be written as a function $f(x) = x^T\beta$ where $\beta \in \mathbf{R}^{d+1}$. The residual sum of squares for $f(x) = x^T\beta$ can be written as

$$\mathrm{R}(\beta) = ||\mathbf{X}\beta - \mathbf{Y}||_2^2. \tag{2.2.9}$$

Using basic calculus, when $\mathbf{X}^T\mathbf{X}$ is nonsingular, the minimum of $\mathrm{R}(\beta)$ occurs at

$$\tilde{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}. \tag{2.2.10}$$

Geometrically, $||\mathbf{X}\beta - \mathbf{Y}||_2$ is the Euclidean distance between $\mathbf{X}\beta$ and $\mathbf{Y}$, thus it is minimized when $\mathbf{X}\tilde{\beta}$ is the projection of $\mathbf{Y}$ on the column space of $\mathbf{X}$.



(a) (b)

Figure 2.1: (a) The scatter plot for 2D linear regression. (b) The scatter plot for nearest neighbor classification.

B.*Interpretation.* A typical loss function for regression is the *quadratic loss*

$$L_{quad}(x, y, h) = (y - h(x))^2. \tag{2.2.11}$$

The Bayes optimal regression function is

$$h_{quad}(x) = \mathrm{E}(y|x). \tag{2.2.12}$$

This can be seen by observing that

$$\begin{aligned}\mathrm{E}[(h(X)-Y)^2] &= \mathrm{E}_X\mathrm{E}_{Y|X}[(h(X)-Y)^2] \\ &= \mathrm{E}_X[(h(X)-\mathrm{E}(Y|X))^2+\mathrm{E}_{Y|X}((Y-\mathrm{E}(Y|X))^2)].\end{aligned}$$

Least squares linear regression is an ERM algorithm with the loss function being the quadratic loss and the hypothesis space being the hyperplanes in $\mathbf{R}^d$.

There is a well-known equivalence between least squares regression and maximum likelihood estimation. Suppose $Y = f(X)+\epsilon$, where $f \in \mathcal{H}$, ϵ is independent of X and $\epsilon \sim N(0,\sigma)$. Let $\epsilon_i = y_i - h(x_i)$, then the joint distribution of $\epsilon_1,\ldots,\epsilon_n$ is given by the pdf $p_h(\epsilon_1,\ldots,\epsilon_n) = \Pi_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\epsilon_i^2}{2\sigma^2}} = \Pi_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i-h(x_i))^2}{2\sigma^2}}$. Thus maximizing the likelihood $p_h(\epsilon_1,\ldots,\epsilon_n)$ over $\{p_h : h \in \mathcal{H}\}$ is the same as minimizing the quadratic loss $\sum_{i=1}^n (y_i - f(x_i))^2$ over $\mathcal{H}$.

C.*Performance guarantee.* The estimate $\tilde{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$ can be shown to converge to the Bayes optimal parameter in general.

The expected risk for a hyperplane $h(\mathbf{x}) = \beta^T\mathbf{x}$ is

$$\begin{aligned}\mathrm{R}(\beta) &= \mathrm{E}[(\beta^T X - Y)^2] \\ &= \mathrm{E}[(\beta^T X)^2 - 2(\beta^T X)Y + Y^2] \\ &= \mathrm{E}(\beta^T X X^T \beta - 2\beta^T XY + Y^2) \\ &= \beta^T \mathrm{E}(XX^T)\beta - 2\beta^T \mathrm{E}(XY) + \mathrm{E}(Y^2).\end{aligned}$$

Assume $\mathrm{E}(XX^T)$ is nonsingular [2], then the minimizer of $\mathrm{R}(\beta)$ is

$$\beta^* = \mathrm{E}(XX^T)^{-1}\mathrm{E}(XY). \tag{2.2.13}$$

Using the law of large numbers, it is easy to show that

$$\tilde{\beta} \xrightarrow{P} \beta^*. \tag{2.2.14}$$

See (Hastie et al., 2005, Chap. 3) for more details on least squares linear regression.

D.*Alternatives.* Certainly, other loss functions may be chosen for regression. An example is the *absolute error loss* $L_1(x, y, h) = |h(x) - y|$. The corresponding optimal regression function is $h_1(x) = \text{median}(Y|x)$. However, quadratic loss is computationally more attractive because it has a closed-form solution, and it is numerically easier to minimize because the minimum can be found numerically using simple gradient descent methods. The latter property is particularly useful when dealing with high-dimensional sparse data.

We can use the density estimation approach for regression as well. We show the following performance bounds in terms of how accurate the density estimate is, as measured by the ℓ_1 distance between the estimate and the true distribution. Recall that for two continuous distributions P_1 and P_2 with pdf's f_1 and f_2 respectively on a random variable Z, the ℓ_1 distance between them is $\int_Z |f_1(z) - f_2(z)|dz$, and is denoted by $||P_1 - P_2||_1$.

Proposition 2. *Let $\tilde{P}(X, Y)$ be an estimation for $P(X, Y)$, $h(x) = \mathrm{E}_{\tilde{P}}(Y|x)$ and*

[2]For nondegenerate $P(X)$, $\mathrm{E}(XX^T)$ is singular iff the components of X are linearly dependent. Let $X = (X_1, \ldots, X_d)$. If $\mathrm{E}(XX^T)$ is singular, then there exists a nonzero vector $\mathbf{c} = (c_1, \ldots, c_d)$ such that $\mathrm{E}(XX^T)\mathbf{c} = \mathbf{0}$, thus $\mathbf{c}^T\mathrm{E}(XX^T)\mathbf{c} = 0$, that is, $\sum_{i,j} c_i c_j E(X_i X_j) = 0$. Since $\sum_{i,j} c_i c_j \mathrm{E}(X_i X_j) = \mathrm{E}(\sum_{i,j} c_i c_j X_i X_j) = \mathrm{E}((\sum_i c_i X_i)^2)$, $\sum_i c_i X_i = 0$ for all $(X_1, \ldots, X_d)$ such that $P(X_1, \ldots, X_d) > 0$. On the other hand, if there exists a nonzero vector $\mathbf{c} = (c_1, \ldots, c_d)$ such that $\sum_i c_i X_i = 0$ for all $(X_1, \ldots, X_d)$ satisfying $P(X_1, \ldots, X_d) > 0$, then it is easy to show that the row vectors of $\mathrm{E}(XX^T)$ are linearly dependent, thus $\mathrm{E}(XX^T)$ is singular.

$h_{quad}(x) = \mathrm{E}_P(Y|x)$. *If* $|Y| \leq C$ *for some constant* C*, and quadratic loss is used, then*

$$\mathrm{R}(h) - \mathrm{R}(h_{quad}) \leq 2C^2 ||P_{X,Y} - \tilde{P}_{X,Y}||_1. \tag{2.2.15}$$

If only an estimate $\tilde{P}(Y|X)$ *is given for* $P(Y|X)$*, then*

$$\mathrm{R}(h) - \mathrm{R}(h_{quad}) \leq 2C^2 \sup_x ||\tilde{P}_{Y|x} - P_{Y|x}||_1. \tag{2.2.16}$$

Proof. By definition, we have

$$\mathrm{R}(h) - \mathrm{R}(h_{quad}) = \mathrm{E}_P((\mathrm{E}_{\tilde{P}}(Y|X) - \mathrm{E}_P(Y|X))^2).$$

Let $f(x,y)$, $\tilde{f}(x,y)$ be the density functions of P and $\tilde{P}$ respectively, then we have

$$\begin{aligned}
&\mathrm{E}_P((\mathrm{E}_{\tilde{P}}(Y|X) - \mathrm{E}_P(Y|X))^2) \\
&= \textstyle\int_X f(x)[\int_Y y\tilde{f}(y|x)dy - \int_Y yf(y|x)dy]^2dx \\
&= \textstyle\int_X f(x)[\int_Y y(\tilde{f}(y|x) - f(y|x))dy]^2dx \\
&\leq \textstyle\int_X f(x)[\int_Y |y||\tilde{f}(y|x) - f(y|x)|dy]^2dx \\
&\leq \textstyle\int_X f(x)C^2[\int_Y |\tilde{f}(y|x) - f(y|x)|dy]^2dx \\
&\leq \textstyle\int_X f(x)[C^2 \int_Y |\tilde{f}(y|x) - f(y|x)|dy \int_Y |\tilde{f}(y|x) + f(y|x)|dy]dx \\
&= \textstyle\int_X f(x)2C^2[\int_Y |\tilde{f}(y|x) - f(y|x)|dy]dx \\
&= 2C^2\mathrm{E}_{X\sim P_X}||\tilde{P}_{Y|X} - P_{Y|X}||_1.
\end{aligned}$$

Now observe that

$$
\begin{aligned}
&\mathrm{E}_{X\sim P_X}||\tilde{P}_{Y|X} - P_{Y|X}||_1 \\
&= \int_{X\times Y} |f(x)\tilde{f}(y|x) - f(x)f(y|x)|dxdy \\
&= \int_{X\times Y} |\tfrac{f(x)}{\tilde{f}(x)}\tilde{f}(x,y) - f(x,y)|dxdy \\
&= \int_{X\times Y} |\tfrac{f(x)-\tilde{f}(x)}{\tilde{f}(x)}\tilde{f}(x,y) + \tilde{f}(x,y) - f(x,y)|dxdy \\
&\leq \int_{X\times Y} |\tfrac{f(x)-\tilde{f}(x)}{\tilde{f}(x)}\tilde{f}(x,y)|dxdy + \int_{X\times Y} |\tilde{f}(x,y) - f(x,y)|dxdy \\
&= \int_X |f(x) - \tilde{f}(x)|dx + ||P_{X,Y} - \tilde{P}_{X,Y}||_1 \\
&\leq 2||P_{X,Y} - \tilde{P}_{X,Y}||_1.
\end{aligned}
$$

Hence we have $\mathrm{R}(h) - \mathrm{R}(h_{quad}) \leq 2C^2||P_{X,Y} - \tilde{P}_{X,Y}||_1$.

For the case when only $\tilde{P}(Y|X)$ is given as an estimate for $P(Y|X)$, note that from the proof above, we have $\mathrm{R}(h) - \mathrm{R}(h_{quad}) \leq 2C^2 E_{X\sim P_X}||\tilde{P}_{Y|X} - P_{Y|X}||_1 \leq 2C^2 \sup_x ||\tilde{P}_{Y|x} - P_{Y|x}||_1$. ■

2.2.3 Nearest Neighbor Classification

A.*Method.* Consider the task of predicting the labels for points in $\mathbf{R}^2$. Assume labeled sample points are shown in Fig 2.1(b). It appears that points with the same label are grouped together. A simple prediction method is to set the label of a test point x to be the majority of the labels for $NN_k(x)$, the set of k training points closest to x.

The method of nearest neighbor (NN) classification can be applied for high dimensional data as well.

B.*Interpretation.* Classification problems generally uses 0/1 loss:

$$L_{01}(x, y, h) = \mathrm{I}(h(x) \neq y) \tag{2.2.17}$$

The Bayes optimal classifier is

$$h_{01}(x) = \arg\max_{y \in Y} P(y|x) \tag{2.2.18}$$

The NN classification rule can be interpreted as a nonparametric density estimation approach, in which the distribution $\tilde{P}(y|x)$ for labels in $NN_k(x)$ as an approximation for $P(y|x)$.

C.*Performance guarantee.* While nearest neighbor estimate is very simple in form, it was shown that if k is allowed to vary with n such that $k(n) \to \infty$ and $k(n)/n \to 0$, then $\mathrm{E}_{(x,y)\sim P}(|\tilde{P}(y|x) - P(y|x)|) \to 0$ irrespective of what P is (Stone, 1977).

We give a general bound for the performance of the density estimation approach for classification.

Proposition 3. *Let $\tilde{P}(X, Y)$ be an estimate for $P(X, Y)$, $h(x) = \arg\max_y P(y|x)$ and $\tilde{h}(x) = \arg\max_y \tilde{P}(y|x)$, then for 0/1 loss,*

$$\mathrm{R}(\tilde{h}) - \mathrm{R}(h) \leq 2||\tilde{P} - P||_1. \tag{2.2.19}$$

Proof. Let $y_x = h(x) = \arg\max_y P(y|x)$ and $\tilde{y}_x = \tilde{h}(x) = \arg\max_y \tilde{P}(y|x)$, then for

each x, we have $P(h(x)|x) \geq P(\tilde{h}(x)|x)$, $\tilde{P}(\tilde{h}(x)|x) \geq \tilde{P}(h(x)|x)$, thus

$$\begin{aligned}
\mathrm{R}(\tilde{h}) - \mathrm{R}(h) &= \textstyle\sum_x \{[1 - P(\tilde{y}_x|x)] - [1 - P(y_x|x))]\} P(x) \\
&= \textstyle\sum_x [P(y_x|x) - P(\tilde{y}_x|x)] P(x) \\
&\leq \textstyle\sum_x \mathrm{I}[y_x \neq \tilde{y}_x][P(y_x|x) - P(\tilde{y}_x|x) + \tilde{P}(\tilde{y}_x|x) - \tilde{P}(y_x|x)] P(x) \\
&= \textstyle\sum_x \mathrm{I}[y_x \neq \tilde{y}_x]\{[P(y_x|x) - P(y_x|x)] + [\tilde{P}(\tilde{y}_x|x) - \tilde{P}(\tilde{y}_x|x))]\} P(x) \\
&\leq \textstyle\sum_x \mathrm{I}[y_x \neq \tilde{y}_x][|P(y_x|x) - \tilde{P}(y_x|x)| + |\tilde{P}(\tilde{y}_x|x) - P(\tilde{y}_x|x))|] P(x) \\
&\leq \textstyle\sum_x ||\tilde{P}_{Y|x} - P_{Y|x}||_1 P(x) \\
&\leq 2||\tilde{P} - P||_1
\end{aligned}$$

■

The above bound is a loose one. In fact, even if $\tilde{P}$ does not converge to P, h and $\tilde{h}$ can still be identical.

D.*Alternatives*. The density estimation method in NN classification does not exploit any knowledge that is particular about the distribution being learned. For example, from Figure 2.1(b), it appears that for each class y, we can assume $P(X|y)$ is a Gaussian distribution with pdf $N(x; \mu, \Sigma)$. Since $P(x, y) = P(y)P(x|y)$, then it just remains to estimate $P(y)$'s, μ_y's, and Σ_y's from the given data. This is Fisher's method of *discriminant analysis*.

2.2.4 Naive Bayes Classifier

A.*Method*. Given a set of training examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, where each observation $\mathbf{x}_i \in A_1 \times \ldots \times A_k$, and each label $y_i \in Y$ for finite sets $A_1, \ldots, A_k, Y$. Assume the data is generated by a distribution $P(A_1, \ldots, A_k, Y)$ satisfying $P(a_1, \ldots, a_k|y) =$

$\Pi_{i=1}^{k} P(a_i|y)$. This implies

$$P(a_1, \ldots, a_k, y) = P(y) \prod_{i=1}^{k} P(a_i|y) \tag{2.2.20}$$

Let $\#(A)$ denote the number of times that event A happens in the training examples. Let $N_y = \#(Y = y)$, $N_{y,i,v} = \#(Y = y, A_i = v)$. From the frequentist interpretation of probability, we should estimate $P(y)$ and $P(a_i|y)$ using

$$\tilde{P}(y) = N_y / \textstyle\sum_{y'} N_{y'}, \tag{2.2.21}$$

$$\tilde{P}(a_i|y) = N_{y,i,a_i} / \textstyle\sum_{a_i'} N_{y,i,a_i'}. \tag{2.2.22}$$

Now given an instance $(a_1, \ldots, a_k)$, predict $\arg\max_{y \in Y} \tilde{P}(y|a_1, \ldots, a_k)$. This is computed by noting that it is the same as $\arg\max_{y \in Y} \tilde{P}(y) \prod_{i=1}^{k} \tilde{P}(a_i|y)$.

B.*Interpretation.* Naive Bayes classifier is an example of parametric density estimation approach. Let p be a distribution on $X \times Y$, the *log loss*

$$L_{log}(x, y, p) \stackrel{\text{def}}{=} -\ln p(x, y) \tag{2.2.23}$$

is commonly used for estimating probability densities because the p with minimal risk is [3]

$$p_{log}(x, y) = P(x, y). \tag{2.2.24}$$

The probability estimation procedure of Naive Bayes is an ERM algorithm in which the loss function is the log loss, and the probability distributions are those satisfying $p(a_1, \ldots, a_k, y) = p(y)\Pi_{i=1}^{k} p(a_i|y)$. Thus each p is parameterized by $p(y)$'s and $p(a_i|y)$'s.

[3] Consider a continuous distribution $P(X)$ on a random variable X with pdf $f(x)$. Let $g(x)$ be an arbitrary pdf for X. Using the inequality $\ln y \leq y - 1$, we have $\mathrm{E}(-\ln g(X)) - \mathrm{E}(-\ln f(X)) = \mathrm{E}(-\ln \frac{g(X)}{f(X)}) \geq \mathrm{E}(-(\frac{g(X)}{f(X)} - 1))) = \int (\frac{g(x)}{f(x)} - 1) f(x) dx = \int (g(x) - f(x)) dx = 1 - 1 = 0$.

We have

$$\begin{aligned} p_n &= \arg\min_p R_n \\ &= \arg\min_p \frac{1}{n} \textstyle\sum_{i=1}^n -\log p(x_i, y_i) \\ &= \arg\max_p \Pi_{i=1}^n p(x_i, y_i) \\ &= \arg\max_p \Pi_{y\in Y} p(y)^{N_y} \Pi_{y\in Y} \Pi_{i=1}^k \Pi_{a_i \in A_i} p(a_i|y)^{N_{y,i,a_i}} \end{aligned}$$

Now the $p(y)$ parameters should be selected to maximize $\Pi_{y\in Y} p(y)^{N_y}$. Note that if $p_1 + \ldots + p_m = 1$, $p_i, f_i \geq 0$ for $1 \leq i \leq m$, with $\sum_{i=1}^m f_i > 0$, then the maximum of $\Pi_{i=1}^m p_i^{f_i}$ occurs when $p_i = \frac{f_i}{\sum_{j=1}^m f_j}$. Thus the maximum of $\Pi_{y\in Y} p(y)^{N_y}$ occurs at $p(y) = \frac{N_y}{\sum_{y'} N_{y'}}$. [4] Similarly we have $p(a_i|y) = \frac{N_{y,i,a_i}}{\sum_{a_i'} N_{y,i,a_i'}}$.

C.*Performance guarantee.* The law of large numbers guarantees that the estimated parameters of Naive Bayes converge in probability to the marginals of the underlying distribution. Using Hoeffding's inequality, we can also derive simple convergence rate.

It should be noted that even though the Naive Bayes assumption is usually not true in practice, it works very well in various applications. This is probably not so surprising after all if one notes that the estimated distribution need not be the same as the true distribution in order to produce the same classification rule.

2.2.5 Domain adaptation

We consider the problem of domain adaptation as another application of statistical learning theory.

Suppose we have two distributions P_s and P_t on $X \times Y$. Let L be an arbitrary loss function. If $P_s(x, y) > 0$ whenever $P_t(x, y) > 0$, then the minimizer h_t of $\mathrm{R}_t(h) =$

[4] by convention, 0^0 is defined to be 1.

$E_{(X,Y)\sim P_t}(L(X,Y,h))$ is the same as the minimizer h_s^w of

$$\mathrm{R}_s^w(h) = E_{(X,Y)\sim P_s}(w(X,Y)L(X,Y,h)),$$

where $w(x,y) = \frac{P_t(x,y)}{P_s(x,y)}$ (with the convention that $w(x,y) = 1$ if $P_s(x,y) = 0$). Hence, with w and training examples from P_s, then one can obtain the minimizer of $E_{(X,Y)\sim P_t}(L(X,Y,h))$.

This result can be useful with additional assumptions that make it possible to estimate w. One such assumption is that $P_s(y|x)$ and $P_t(y|x)$ are identical. In this case, $w(x,y) = \frac{P_t(x)}{P_s(x)}$. Estimating $P_t(x)$ and $P_s(x)$ is generally a simpler problem than learning the joint densities in practice. For example, with sufficiently many samples, the neighborhood sizes can be used as density estimates. Alternatively, more sophisticated density models may be estimated independently from data. For example, in natural language processing, these densities may be estimated based on some language models.

Another assumption that can simplify w is that $P_s(x) = P_t(x)$. In this case, $w(x,y) = \frac{P_t(y|x)}{P_s(y|x)}$. However, estimating w by estimating $P_t(y|x)$ and $P_s(y|x)$ is not useful and sound in practice, because if $P_t(y|x)$ can be estimated accurately, it can be used for prediction directly; if it cannot be estimated accurately, then it is unlikely that using it to estimate other quantities can lead to good performance with any guarantee.

If P_s and P_t are very similar, then it may be better to just use h_s as an approximation for h_t, rather than trying to learn w. Suppose the loss function is bounded by C, and let $\epsilon = ||P_s - P_t||_1$. Then the minimizer h_s of $E_{(X,Y)\sim P_s}(L(X,Y,h))$ is at most $2C\epsilon$ worse than h_t on P_t, because $\mathrm{R}_t(h_s) - \mathrm{R}_t(h_t) = \mathrm{R}_t(h_s) - \mathrm{R}_s(h_s) + \mathrm{R}_s(h_s) - \mathrm{R}_t(h_t) \leq \mathrm{R}_t(h_s) - \mathrm{R}_s(h_s) + \mathrm{R}_s(h_t) - \mathrm{R}_t(h_t) = 2C||P_s - P_t||_1$.

2.3 Components of Learning Machines

In general terms, a learning system usually consists of the following components:

(a) Representation: Choose a representation function r which converts an observation X to $r(X)$. Here an observation is simply what is available to a learning system, and can be the raw data or some preprocessed data. r is any transformation which converts X in a form the system will deal with, which is generally a mapping from X to real vectors.

(b) Approximation: Choose a hypothesis space $\mathcal{H}$ which is likely to contain the true law between $r(X)$ and Y. Depending on the learning problem, the hypothesis space $\mathcal{H}$ can contain very different objects. For example, in density estimation $\mathcal{H}$ contains distributions, while in regression, it contains functions.

(c) Learning/Estimation: Use the training data to obtain knowledge about the optimal hypothesis between $r(X)$ and Y. Examples of what we mean by knowledge include estimated densities in the density estimation approach, and estimated decision rules in the ERM approach.

(d) Prediction: Use the knowledge learned to infer values of unknown variables. For example, using estimated densities or decision rules for predicting labels in classification.

The performance of a system can be analyzed in terms of the error of each component. Consider using the risk minimization approach to learn a prediction rule mapping observation X to outcome Y. Let $R^* = \arg\min_h \mathrm{E}(L(h(X), Y))$, where h ranges over arbitrary functions from X to Y. Let $R_r^* = \arg\min_h \mathrm{E}(L(h(r(X)), Y))$, where h ranges over arbitrary functions from $r(X)$ to Y. Let $R_{r,\mathcal{H}}^* = \arg\min_{h\in\mathcal{H}} \mathrm{E}(L(h(r(X)), Y))$. Let $R_n = \mathrm{E}(L(h_n(r(X)), Y))$, where h_n is the hypothesis output by the learner on n i.i.d. training examples. Let $\tilde{R}_{n,m}$ be the empirical risk of h_n on m i.i.d. test examples. The error of a learning algorithm can be decomposed as follows:

(a) Representation error $R_r^* - R^*$.

(b) Approximation error $R_{r,\mathcal{H}}^* - R_r^*$.

(c) Learning/Estimation error $R_n - R_{r,\mathcal{H}}^*$.

(d) Prediction variation $\tilde{R}_{n,m} - R_n$.

We illustrate the above errors using a simple example.

Example 4. Suppose the underlying distribution is

$$\begin{aligned} P(X_1, X_2, Y) &= P(X_1, X_2)P(Y|X_1, X_2), \text{ where} \\ P(X_1, X_2) &= U([-a, a] \times [-b, b]), \\ P(Y|X_1, X_2) &= \mathrm{I}(Y = f(X_1, X_2) = 1 + X_1 + X_1^2 + X_2). \end{aligned}$$

Consider regression of Y against X using quadratic loss. We have $R^* = 0$ because $P(Y|X_1, X_2)$ is deterministic.

If the representation function is chosen as $r(X) = X_1$, then the minimizer corresponding to R_r^* is $h_r^*(X_1) = \mathrm{E}(Y|X_1) = 1 + X_1 + X_1^2$. It follows that

$$\begin{aligned} R_r^* &= \int_{-a}^{a}\int_{-b}^{b}(y - 1 - x_1 - x_1^2)^2 \tfrac{1}{4ab} dx_2 dx_1 \\ &= \int_{-a}^{a}\int_{-b}^{b} x_2^2 \tfrac{1}{4ab} dx_2 dx_1 \\ &= \frac{b^2}{3} \end{aligned}$$

The representation error is thus $R_r^* - R^* = \frac{b^2}{3}$.

Now consider only linear functions of $r(X)$ as hypotheses. The expected risk for $h(X_1) = \alpha + \beta X_1$ is given by

$$\begin{aligned} & \int_{-a}^{a}\int_{-b}^{b}(y - \alpha - \beta x_1)\tfrac{1}{4ab} dx_2 dx_1 \\ = \; & \int_{-a}^{a}\int_{-b}^{b}(1 + x_1 + x_1^2 + x_2 - \alpha - \beta x_1)^2 \tfrac{1}{4ab} dx_2 dx_1 \\ = \; & \frac{a^4}{5} + \frac{a^2}{3}(1 - \beta)^2 + (1 - \alpha)^2 + \frac{2a^2}{3}(1 - \alpha) + \frac{b^2}{3}. \end{aligned}$$

The minimum occurs at $(\alpha, \beta) = (1, 1)$. Thus $R_{r,\mathcal{H}}^* = \frac{a^4}{5} + \frac{b^2}{3}$, and the approximation

error is $R^*_{r,\mathcal{H}} - R^*_r = \frac{a^4}{5}$.

The estimation error is a random variable. It is possible to compute its distribution, but we shall skip the tedious computations here. □

In the next few sections, we discuss the problems involved and techniques used in representation, approximation, estimation and prediction.

2.3.1 Representation

In practice, an observation X is often mapped to an *attribute vector* $(r_1(X), \ldots, r_d(X))$, where each attribute $r_i(X)$ takes real values, categorical values (ordered or unordered), or binary values. A more general representation scheme considers *features*, which are functions of X and Y, used to explicitly capture the correlation between X and Y. Note that in the literature, features are often used to refer to attributes as well. What we call feature here is also often called a joint feature map in the literature (Tsochantaridis et al., 2004). We shall use these two terms strictly in the senses defined here, that is, attributes are functions of observations only, while features are functions of the observation and the output.

Effectiveness of a representation is domain-dependent, and is a major component in applied machine learning. For example, the TF-IDF representation (Spärck Jones, 1972) is empirically effective for text classification and have also been justified in various theoretical models (Robertson, 2004). In addition, effectiveness of a representation depends on the hypothesis space chosen as well. For example, when learning functions of the form $f(x) = a \sin x + be^x + ce^{-x}$ where a, b, c are parameters, if x is represented as $(\sin x, e^x, e^{-x})$, then linear regression suffices, but the representation x alone is not useful for linear regression.

As a rule of thumb, it is wise to include any information which may be relevant in the representation first. Not including relevant information lowers the up-

per bound on the performance of a system, while including irrelevant information generally can be overcome by other means in later steps. For example, with sufficiently many data, the effect of irrelevant information such as noises in the observation, redundant attributes or features, can generally be overcome. Formally, consider random noises X' generated independently of the original observation X, then $\min_h \mathrm{E}(L(h(X, X'), Y)) = \min_h \mathrm{E}(L(h(X), Y))$. But including irrelevant information comes with the price of potential overfitting, especially when there is little data. For example, suppose $Y = \beta X + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$. With n additional random attributes, training on n examples can always be done with 0 empirical loss, no matter how Y is distributed. This means that the coefficients can be far from the optimal. Thus the convergence can be arbitrarily delayed using more irrelevant random attributes.

Once an initial set of potentially useful features is chosen, a more effective representation can be constructed in various ways, such as subset selection (Guyon and Elisseeff, 2003), basis expansion, the kernel trick, principal component analysis (Pearson, 1901), multidimensional scaling (Bronstein et al., 2006), Isomap (Tenenbaum et al., 2000), locally linear embedding (Roweis and Saul, 2000). We briefly describe some of these methods below.

Subset selection is generally done to remove irrelevant features, which can easily cause overfitting when little data is available. For computational reasons, this is often done greedily by either induction or filtering. Induction starts with an empty set of attributes or features, and augments it by repeatedly adding promising ones from the candidate sets. Filtering starts with all candidates, and repeatedly removes those that can be dropped without hurting performance. A comprehensive introduction to feature selection can be found in (Guyon and Elisseeff, 2003). A voting algorithm to select relevant features and examples is given in (Blum and Langley, 1997). Efficient feature induction algorithms have been studied for various models like maximum entropy models (Berger et al., 1996), Markov random fields (Della Pietra et al., 2002), Conditional

random fields (McCallum, 2003).

Basis expansion transforms an observation $\mathbf{x}$ to an attribute vector $(h_1(\mathbf{x}), \ldots, h_m(\mathbf{x}))$, where each h_i is prespecified and called a basis function. Such input transform is equivalent to transformations on the hypothesis space. For example, consider regression on $(\mathbf{x}, y)$ pairs from $\mathbf{R}^d \times \mathbf{R}$. Consider the basis functions $h_i(\mathbf{x}) = x_i$ for $1 \leq i \leq d$, and $h_{ij}(\mathbf{x}) = x_i x_j$, then performing linear regression using the new basis representation is equivalent to using quadratic functions as the hypothesis space.

The *kernel trick* is applied to algorithms which require the evaluation of inner products of instances only. It replaces the inner product $\langle x_i, x_j \rangle = x_i^T x_j$ by some other inner product $k(x_i, x_j)$. k is usually called a *kernel function*. Classic examples include kernel PCA and SVMs. See (Schölkopf and Smola, 2002) for more details.

Principal component analysis (PCA) (Pearson, 1901) looks for directions along which maximum amount of data variation can be explained, and then project data into these directions, in the hope that irrelevant information will be throwed away. Formally, given points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbf{R}^d$, the variance of the projections of them on a unit vector $\mathbf{v}$ is given by

$$\frac{1}{n} \sum_{i=1}^n [\mathbf{v}^T (\mathbf{x}_i - \overline{\mathbf{x}})]^2 \quad = \quad \mathbf{v}^T \Sigma \mathbf{v}$$

where $\Sigma = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T$ is the empirical covariance matrix. Thus we want to maximize $\mathbf{v}^T \Sigma \mathbf{v}$ subject to the constraint that $||\mathbf{v}||_2^2 - 1 = 0$. The Lagrangian and its derivative with respect to $\mathbf{v}$ are

$$\begin{aligned} L(\mathbf{v}, \lambda) &= \mathbf{v}^T \Sigma \mathbf{v} - \lambda(||\mathbf{v}||_2^2 - 1) \\ \nabla_{\mathbf{v}} L &= 2\Sigma \mathbf{v} - 2\lambda \mathbf{v} \end{aligned}$$

Thus $\mathbf{v}$ is an eigenvector of Σ. In addition, $\mathbf{v}^T \Sigma \mathbf{v} = \mathbf{v}^t \lambda \mathbf{v} = \lambda$. Hence the directions which explain most variations of the data are the eigenvectors corresponding to the

largest eigenvalues.

2.3.2 Approximation

Expressiveness and tractability are two key factors involved in choosing or designing hypothesis spaces. Expressiveness is often used to refer to the variety of hypotheses included in the hypothesis space. A hypothesis space should be chosen such that it is expressive enough to account for reality, but not too expressive that it is impossible to infer which hypothesis is most likely to produce the training data. Tractability is whether the model can be efficiently learned and applied for prediction.

The empirical measure on expressiveness of a hypothesis space is that it should be able to fit the training data reasonably well, assuming the training data contains sufficient information for a hypothesis that fits well on it to be defined.

The theoretical measure is much more involved. The key idea is that expressiveness should be measured by the projection of the hypothesis space on finite samples. If given any set of observations $x_1, \ldots, x_n$, and for any possible sequence of outputs $y_1, \ldots, y_n$, there exists a hypothesis that fits the training data well, then the hypothesis space is able to fit noise well. Such a hypothesis is expected to be too expressive, and it is hard to obtain any performance guarantee. We shall make this precise in next section.

Another kind of expressiveness required is the ease of integrating domain knowledge in the model. For example, it is easy to incorporate arbitrary dependencies in discriminative models like maximum entropy models, while it is hard to do so in generative models like Naive Bayes.

When choosing or designing a hypothesis space, generally there is a tradeoff in expressiveness and tractability. An expressive model is often associated with computational difficulties in learning and prediction, and strong assumptions on the targets have to be made in order to make learning and prediction tractable. For example, naive Bayes classifier avoids modeling all the observations by making independence

assumptions which allow the model to be decomposed into a small number of factors, where each factor can be estimated separately in reasonable accuracy even with a small number of examples.

Certainly, there are other considerations when choosing and designing a hypothesis space. For example, interpretability may be required in some situations, and we also want to avoid models requiring heavy engineering, such as heavy tuning of model parameters.

2.3.3 Estimation

In this section, we discuss learning techniques which have guarantee on performance measures such as generalization error, stability to perturbation in data, and computational efficiency.

A.*ERM and generalization.* As we have seen, ERM is a generic learning algorithm that includes many learning algorithms as special instances. ERM may be informally justified as follows: for a hypothesis h, as more i.i.d. training examples become available, h's empirical risk on the sample should converge to its true risk, by the law of large numbers. Thus minimizing empirical risk over a large sample is expected to be a good approximation as minimizing the true risk. However, it should be noted that the above intuition breaks down when the hypothesis space includes all possible functions. Hence the hypothesis space cannot be too expressive. Vapnik (1998) introduced a measure on the expressiveness of a hypothesis space, and obtained a very general sufficient condition for ERM to be consistent. We describe Vapnik's result for 0/1 loss below. For more on statistical learning theory, see (Vapnik, 1999) for a concise overview, (Vapnik, 1995) for a more detailed overview and (Vapnik, 1998) for technical details.

Let $\mathcal{H}$ be a set of functions from X to Y. Each $h \in \mathcal{H}$ corresponds to a function $g_h(x, y) = L(h(x), y)$. Note that if P is the true distribution on $X \times Y$, then $\mathrm{R}(h) =$

$\mathrm{E}_{(x,y)\sim P}(g_h)$. The set $\mathcal{G} = \{g_h : h \in \mathcal{H}\}$ is called the loss class for $\mathcal{H}$. The projection of $\mathcal{G}$ on data points $D = \{z_1, \ldots, z_n\} \subseteq X \times Y$ is $\mathcal{G}_D = \{(g(z_1), \ldots, g(z_n)) : g \in \mathcal{G}\}$. If 0/1 loss is used, then $\mathcal{G}_D$ is a finite set containing at most 2^n elements, and $N(z_1, \ldots, z_n) \stackrel{\text{def}}{=} |\mathcal{G}_D|$ measures how rich the $\mathcal{G}$ is on D. $H(n) = \mathrm{E}(\ln N(z_1, \ldots, z_n))$ is called the *VC entropy*.

Theorem 5. *(Vapnik, 1998) If* $\lim_{n\to\infty} \frac{H(n)}{n} = 0$*, then ERM is consistent. That is, if* h_n *is the hypothesis produced by ERM on* n *training examples, and* h^* *is the minimum risk hypothesis in* $\mathcal{H}$*, then* $\mathrm{R}(h_n) \xrightarrow{P} \mathrm{R}(h^*)$.

Consistency does not imply fast convergence rate. In order to guarantee fast convergence rate, a more stringent criterion is needed. Let $H_{ann}(n) = \ln \mathrm{E}(N(z_1, \ldots, z_n))$, then $H_{ann}(n) \geq H(n)$. $H_{ann}(n)$ is called the *annealed VC entropy*.

Theorem 6. *(Vapnik, 1998)* $\mathrm{P}(\sup_h |R(h) - R_n(h)| > \epsilon) \leq 4 \exp((\frac{H_{ann}(2n)}{n} - \epsilon^2)n)$.

Thus if $\lim_{n\to\infty} \frac{H_{ann}(n)}{n} = 0$, then ERM has a fast convergence rate.

A distribution independent result on convergence rate can be derived from the above result. Define the *growth function* of $\mathcal{H}$ as

$$G(n) = \ln \sup_{z_1,\ldots,z_n} N(z_1, \ldots, z_n). \tag{2.3.1}$$

Then $G(n) \geq H_{ann}(n)$ for any distribution, thus it follows that

Theorem 7. *(Vapnik, 1998)* $\mathrm{P}(\sup_h |R(h) - R_n(h)| > \epsilon) \leq 4 \exp((\frac{G(2n)}{n} - \epsilon^2)n)$.

In addition, the condition

$$\lim_{n\to\infty} \frac{G(n)}{n} = 0, \tag{2.3.2}$$

is a necessary and sufficient condition for consistency of ERM for any distribution.

For the density estimation approach for learning, it should be noted that while consistency provides theoretical guarantee for convergence to the best model in the

hypothesis space, it is not always necessarily to be consistent. For example, the DOP estimation algorithm for constructing parse trees is biased and inconsistent, but works pretty well in practice (Johnson, 2002). Wainwright analyzed a case in which an inconsistent method is provably beneficial (Wainwright, 2006).

B.*Regularization and stability.* Hadamard considered a problem as meaningful only if it is well-posed. To be precise, take a problem as a binary relation between two sets. In the case that (U, ρ_U) and (Z, ρ_Z) are two metric spaces, then R is a *well-posed* problem on the pair (U, ρ_U), (Z, ρ_Z) if R is a subset of $U \times Z$ such that

(a) (Existence) For any $u \in U$, there exists at least one z such that uRz,

(b) (Unique) For any $u \in U$, there is at most one z such that uRz, and

(c) (Stability) For any $\epsilon > 0$, there exists $\delta > 0$ such that whenever $\rho_U(u_1, u_2) < \delta$, it follows that $\rho_Z(z_1, z_2) < \epsilon$ for any z_1, z_2 such that $u_1 R z_1$ and $u_2 R z_2$.

(U, ρ_U) represents the problem space, and (Z, ρ_Z) represents the solution space.

However, it is found later that many natural problems are ill-posed. In particular, the problem of density estimation is ill-posed. When a problem is ill-posed because of violation of (b) or (c), Tikhonov introduced the regularization technique to perturb an ill-posed problem to a well-posed problem (Tikhonov, 1963). The technique can be used to obtain a solution that can be made arbitrarily close to a solution of the original problem.

Example 8 (Regularized linear regression)**.** For least squares linear regression, when the attributes are highly correlated, $\mathbf{X}^T\mathbf{X}$ is close to a singular matrix, and the estimator $\tilde{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$ is unstable. In addition, if one attribute is the linear combination of others, then $\mathbf{X}^T\mathbf{X}$ is singular and the formula is no longer valid, although $\mathrm{R}(\beta)$ still has a minimum.

Tikhonov reformulated the problem by minimizing

$$\mathrm{R}_\lambda(\beta) = \mathrm{R}(\beta) + \lambda\beta^T\beta, \tag{2.3.3}$$

where $\lambda > 0$ is a constant. The minimum occurs at

$$\tilde{\beta}_\lambda = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{Y}. \tag{2.3.4}$$

The RHS is always defined because $\mathbf{X}^T\mathbf{X} + \lambda I$ is nonsingular. [5] In addition, if β^* is a minimizer of $\mathrm{R}(\beta)$, then we have $\mathrm{R}(\beta^*) \leq \mathrm{R}(\tilde{\beta}_\lambda) < \mathrm{R}_\lambda(\tilde{\beta}_\lambda) \leq \mathrm{R}_\lambda(\beta^*) = \mathrm{R}(\beta^*) + \lambda\beta^{*T}\beta^*$. Hence, when $\lambda \to 0$, $\mathrm{R}_\lambda(\tilde{\beta}_\lambda) \to \mathrm{R}(\beta^*)$. □

Certain seemingly ad hoc methods can be interpreted as regularization. For example, it is well-known that initializing the event counts when estimating parameters in naive Bayes can be interpreted as regularization.

Example 9 (Regularized naive Bayes classifier)**.** In naive Bayes classifier, given an instance $(a_1, \ldots, a_k)$, if some $\tilde{P}(a_i|y) = 0$, then $P(y|a_1, \ldots, a_k) = 0$, and thus the instance will not be predicted to be in class y no matter how strong the other attributes suggest that the instance is in class y. In addition, when $P(y)$ is small, underestimation of $P(y)$ and $P(a_i|y)$'s can make it very unlikely to classify an instance in class y. This motivates using $\frac{N_{y,i,a_i}+\alpha}{\sum_{a_i'}[N_{y,i,a_i'}+\alpha]}$ as the estimator for $P(a_i|y)$ and $\frac{N_y+\alpha}{\sum_{y'}[N_{y'}+\alpha]}$ as the estimator for $P(y)$, where $\alpha > 0$ is a constant. The above method of initializing the counts by c is called Laplace correction.

It is easy to verify that Laplace correction corresponds to adding the regularization term $-\log\prod_y p(y)\prod_y\prod_i\prod_{a_i} p(a_i|y)$ to the empirical log loss $\frac{1}{n}\sum_{i=1}^n -\log p(x_i, y_i)$. Note that this term is smaller when the class distribution $p(Y)$ and the attribute value

[5] Assume otherwise, then there exists nonzero vector $\mathbf{c}$ such that . $(\mathbf{X}^T\mathbf{X} + \lambda I)\mathbf{c} = \mathbf{0}$, thus $\mathbf{c}^T(\mathbf{X}^T\mathbf{X}+\lambda I)\mathbf{c} = 0$. However, $\mathbf{c}^T(\mathbf{X}^T\mathbf{X}+\lambda I)\mathbf{c} = (\mathbf{X}\mathbf{c})^T(\mathbf{X}\mathbf{c})+\lambda\mathbf{c}^T\mathbf{c} \geq 0+\lambda\mathbf{c}^T\mathbf{c} > 0$, a contradiction.

distributions $P(A_i|y)$'s are closer to uniform. So while the empirical log loss represents preference over models which fits the data well, the new term represents preference over uniform distributions.

The above regularization term can be seen as a special case of using a Dirichlet distribution as the prior on the parameters. Let p_i denote the probability of the i-th class, and $p_{y,i,j}$ denote the probability that the i-th attribute takes its j-th possible value given that the class is y. Assume these parameters follow Dirichlet distributions with the density functions

$$\begin{aligned} f(p_1, \ldots, p_c; \alpha_1, \ldots, \alpha_c) &= B(\alpha_1, \ldots, \alpha_c) \prod_{i=1}^{c} p_i^{\alpha_i - 1} \\ f(p_{y,i,1}, \ldots, p_{y,i,n_i}; \alpha_{y,i,1}, \ldots, \alpha_{y,i,n_i}) &= B(\alpha_{y,i,1}, \ldots, \alpha_{y,i,n_i}) \prod_{j=1}^{c} p_{y,i,j}^{\alpha_{y,i,j} - 1}, 1 \leq i \leq k \end{aligned}$$

Then it is clear that Laplace correction corresponds to the case when all α_i values are set to $\alpha + 1$. □

In general, ERM can be regularized by minimizing

$$\mathrm{R}_n(h) + \frac{\lambda r(h)}{n} \tag{2.3.5}$$

instead of $\mathrm{R}_n(h)$, where $r(h)$ is a suitable nonnegative function of h, $\lambda > 0$ a constant. As n tends to infinity, the regularization term vanishes and regularized ERM converges to ERM. There are several ways to interpret regularization.

(a) Regularization can be interpreted as trading off the complexity of a hypothesis and its quality of fit. In this case $r(h)$ is treated as a complexity measure.

(b) A related interpretation is that regularization can be treated as searching for the best-fitting hypothesis among hypothesis with a bounded complexity. This is an application of Lagrangian duality theory: There exists a non-increasing function

$s(\lambda)$ such that for any $\lambda > 0$, $\min_h \mathrm{R}_n(h) + \frac{\lambda r(h)}{n}$ is the same as $\min_{r(h) \leq s(\lambda)} \mathrm{R}_n(h)$.

(c) In some cases, regularization can also be interpreted in the Bayesian framework as specifying a prior distribution on the hypothesis, and searching for the maximum a posteriori (MAP) hypothesis. To be precise, it is assumed that $P((x,y)|h) \propto e^{-L(h(x),y)}$ and $P(h) \propto e^{-\lambda r(h)}$. Thus the posterior is given by

$$P(h|(x_1,y_1),\ldots,(x_n,y_n)) \;=\; \frac{P(h)\prod_{i=1}^n P((x_i,y_i)|h)}{P((x_1,y_1),\ldots,(x_n,y_n))} \propto e^{-\lambda r(h) - \sum_{i=1}^n L(h(x),y)}$$

Hence maximizing $P(h|(x_1,y_1),\ldots,(x_n,y_n))$ is the same as minimizing $\mathrm{R}_n(h) + \frac{\lambda r(h)}{n}$.

Stability of regularized ERM implies that the learned hypothesis is insensitive to perturbation or irregularity in data. In practice, regularization is useful for combating overfitting – a situation in which the selected hypothesis fits the training data so well that its performance on test data is not as good as a hypothesis that fits less well on training data.

C.*Smooth Approximations and efficiency.* In practice, ERM can be intractable (Feldman et al., 2009). In such cases, it is useful to minimize a smooth (continuous and differentiable to some order) approximation instead. Besides smoothness, it is desirable that such approximations are convex, or bound the empirical risks.

Example 10 (Classification as regression)**.** Suppose $Y = \{-1, 1\}$ and $X = \mathbf{R}^{d+1}$, where the first coordinate of the observation is 1. Consider using linear functions as classifiers, that is, each classifier h is of the form $h(x) = w^T\mathbf{x}$. In this case, finding an hyperplane with minimal 0/1 loss is intractable.

If we consider the problem as a regression problem, then we can train with quadratic loss. However, using square loss can cause the estimation error on the training data to be arbitrarily close to $\frac{1}{2}$ even if the training data is separable. To see this, suppose the training data consists of $(-x, -1)$, n copies of $(-1, -1)$, n copies of $(1, 1)$ and $(x', 1)$.

least squares parameter is given by $\beta^T = \frac{x+x'+2n}{(2n+1)(x^2+x'^2)+2xx'+4n(n+1)} \begin{pmatrix} x - x' \\ 2n+2 \end{pmatrix}$. Choose positive x and x' such that $x' > x + 2n + 2$, then $(-x, -1)$, $(-1, -1)$ and $(x', 1)$ are classified correctly, while $(1, 1)$ is not. Thus the error is $\frac{n}{2n+1}$.

A better loss function is the exponential loss. Note that h makes a mistake on $\mathbf{x}_i$ iff $y_i h(\mathbf{x}_i) < 0$. Thus the classification error is $\mathrm{R}_n(h) = \sum_{i=1}^n \mathrm{I}(y_i h(\mathbf{x}_i) < 0) \leq \sum_{i=1}^n e^{-y_i h(\mathbf{x}_i)} = \sum_{i=1}^n e^{-y_i w^T \mathbf{x}_i}$. Thus $\hat{\mathrm{R}}_n(h) = \sum_{i=1}^n e^{-y_i w^T \mathbf{x}_i}$ is an upper bound of $\mathrm{R}_n(h)$. Compared to $\mathrm{R}_n(h)$, $\hat{\mathrm{R}}_n(h)$ has the nice property of being smooth, and can be easily minimized using standard optimization algorithms. In practice, using exponential loss works well. □

Note that the second method minimizes an upper bound of the original objective.

We conclude this section by pointing out that the technique of smooth approximation is what makes efficient training of neural networks possible. Rosenblatt's perceptron consisted of a network of linear unit, but training was only done for the last output unit. It was only in the 1970's, Werbos (1974) proposed an efficient training method, the backpropagation algorithm, which replaces the thresholded output of each internal unit by a smooth non-linear function of the unthresholded output.

2.3.4 Prediction

The risk minimization approach generally learns a prediction rule which is easy to evaluate. For the density estimation and the Bayesian approach, making decisions using Eq. 2.2.1 can be computationally challenging because the required probability values can be hard to compute (which is generally true for the Bayesian approach), or the loss may not be decomposable. In this case approximate inference methods or surrogate loss functions or simple decision rules are used instead. Chapter 6 discusses a particular non-decomposable loss, F-measures, for which efficient algorithms exist.

2.4 The Role of Prior Knowledge

The learning algorithms presented so far are designed to work for scenarios following certain assumptions, and it is expected that when the assumptions are violated, they may perform poorly. For example, least squares linear regression relies on the assumption that the relationship between X and Y is approximately linear, and Naive Bayes classifier relies on the independence of the attributes. These assumptions are made after observing some sample data or need to be explicitly given as prior knowledge.

Is it possible to design a machine learning algorithm that are guaranteed to generalize well on arbitrary data?

Consider a simple example. Suppose $X = \{1, 2, \ldots, N\}$, and $Y = \{0, 1\}$ and the objective is to learn an unknown target function f from X to Y. If the following training data is observed $(1, 0), (2, 1), (3, 0), (4, 1), (5, 0), (6, 1)$, then we are likely to guess that the function is $f(x) = \mathrm{I}(x \text{ is even})$. However, based on the training data alone, all consistent functions from X to Y are not distinguishable on unseen data. To be precise, assume that any of the 2^N functions from X to Y is equally likely to be the true function, and suppose the training data consists of $(x_1, y_1), \ldots, (x_n, y_n)$, then for any consistent function h from X to Y, the expected number of errors on unseen x's is a constant. That is, if $h(x_1) = f(x_1), \ldots, h(x_n) = f(x_n)$, then $\mathrm{P}(h(x) \neq f(x) | x \notin \{x_1, \ldots, x_n\})$ is a constant independent of h.

The above simple example thus demonstrates that in some sense generalization beyond training data necessarily requires some prior knowledge. There are many results which support the insight that for various measures of generalization, without any knowledge on the underlying distribution, it is impossible to design a learning algorithm which necessarily generalizes well using observed data only. These results are often called No-Free-Lunch (NFL) theorems.

2.4.1 NFL for Generalization beyond Training Data

Wolpert investigated how a learner can generalize beyond training data for the case where there are finitely many possible observations and outputs, and proved NFL theorems which demonstrate that without prior knowledge, all learners are indistinguishable in terms of generalization ability beyond training data (Wolpert, 1996a). The learning problem considered by Wolpert is as follows.

For learning, the learner is presented with a sequence D consisting of (X, Y) pairs generated i.i.d. from $P(X, Y) = \pi(X)f(Y|X)$, where X and Y are the observation and output variables respectively, both having finitely many possible values, $\pi(X)$ is a fixed (but unknown) distribution, and $f(Y|X)$ is an unknown conditional distribution. The set of all possible f is thus the Cartesian product of $|X|$ unit simplexes.

For prediction, the learner is presented with queries drawn i.i.d. from $\pi(X)$, filtering queries observed in D – thus the queries are *off-training-set* (OTS) queries. In addition, given an instantiation d of D, the queries follow the distribution $P(q|d) = \mathrm{I}(q \notin d_X)\pi(q)/\sum_{q' \notin d_X} \pi(q)$, where d_X is the set of X values appeared in d. The true output y_F for a query q follows the distribution $P(y_F|f, q) = f(y_F|q)$.

A learner $\mathcal{L}$ is modeled as a probability distribution $P_{\mathcal{L}}(h|d)$, where h is a distribution on $X \times Y$. The output y_H of $\mathcal{L}$ on a query $q \in X$ can be represented using a distribution $P_{\mathcal{L}}(y_H|d, q)$.

Intuitively, the learner's objective is to minimize the average-case error. The error on a single prediction is measured using a loss function $L(y_H, y_F)$, where y_H is the predicted output, and y_F is the true output generated by the true law f. For a fixed f, an algorithm's ability to generalize beyond training data can be represented by the distribution $P_{\mathcal{L}}(c|f, d)$ of its loss c. One possible formulation of the average-case loss is to use $P_{\mathcal{L}}(c|f, d)$ averaged over all possible f, that is $\int_f P_{\mathcal{L}}(c|f, d)df / \int_f df$. This average gives equal weights to all f, and is thus called the *uniform f-average* of $P_{\mathcal{L}}(c|f, d)$.

If L is the 0/1 loss, then the uniform f-average of $P_{\mathcal{L}}(c|f, d)$ is $\delta(c, 1)\frac{|Y|-1}{|Y|}$, where

$\delta(a,b) = \mathrm{I}(a=b)$ is the Dirac delta function. First note that

$$\begin{aligned} P(c|f,d) &= \textstyle\sum_{q,y_H,y_F} P_{\mathcal{L}}(c,y_H,y_F,q|f,d) \\ &= \textstyle\sum_{q,y_H,y_F} P(q|f,d)P(y_F|f,d,q)P(y_H|f,d,q,y_F)P(c|f,d,q,y_F,y_H) \\ &= \textstyle\sum_{q,y_H,y_F} P(q|d)P(y_F|f,q)P_{\mathcal{L}}(y_H|d,q)\delta(c,L(y_F,y_H)) \end{aligned} \tag{2.4.1}$$

From the above formula, we have

$$\begin{aligned} \textstyle\int_f P_{\mathcal{L}}(c|f,d)df &= \textstyle\int_f \sum_{q,y_H,y_F} P(q|d)P(y_F|f,q)P(y_H|d,q)\delta(c,L(y_F,y_H))df \\ &= \textstyle\sum_{q,y_H,y_F} P(q|d)P(y_H|d,q)\delta(c,L(y_F,y_H)) \int_f P(y_F|f,q)df \end{aligned} \tag{2.4.2}$$

Consider $s(y_F,q) = \int_f P(y_F|f,q)df$. It is symmetric in the y_F values, and the q values, thus s has the same value for all (y_F,q) pairs. In addition, $\sum_{y_F} s(y_F,q) = \int_f df$. Thus $s(y_F,q) = \frac{1}{|Y|}$. Hence

$$\begin{aligned} \textstyle\int_f P_{\mathcal{L}}(c|f,d)df &= \textstyle\sum_{q,y_H,y_F} P(q|d)P(y_H|d,q)\delta(c,L(y_F,y_H)) \int_f df \\ &= \textstyle\sum_q P(q|d) \sum_{y_H} P(y_H|d,q) \sum_{y_F} \delta(c,L(y_F,y_H)) \int_f df \\ &= \delta(c,1)\tfrac{|Y|-1}{|Y|} \textstyle\int_f df \end{aligned}$$

In general, if there exists a function Λ such that $\Lambda(c) = \sum_{y_F} \delta(c,L(y_F,y_H))$ for any y_H, then the uniform f-average of $P_{\mathcal{L}}(y_F|f,d)$ is $\frac{\Lambda(c)}{|Y|}$. This can be shown by following the above proof closely.

Theorem 11. *(Wolpert, 1996a) Suppose there exists Λ such that $\Lambda(c) = \sum_{y_F} \delta(c,L(y_F,y_H))$ for any y_H. For any learner $\mathcal{L}$, the uniform f-average of $P_{\mathcal{L}}(c|f,m)$ is $\Lambda(c)/|Y|$.*

Proof. First note that $P_{\mathcal{L}}(c|f,m) = \sum_d P_{\mathcal{L}}(c,d|f,m) = \sum_d P_{\mathcal{L}}(d|f,m)P_{\mathcal{L}}(c|f,d,m) = \sum_d P_{\mathcal{L}}(d|f,m)P_{\mathcal{L}}(c|f,d)$.

For any f and d, split f into f_{d_X} consisting of all $f(Y|x)$'s for $x \in d_X$, and $f_{\sim d_X}$

consisting of $f(Y|x)$'s for $x \notin d_X$. The assumptions in the generation of d and the generation of the true outputs on queries imply that $P_{\mathcal{L}}(d|f,m)$ depends only on f_{d_X} and $P_{\mathcal{L}}(c|f,d)$ only depends on $f_{\sim d_X}$. The proof for $\int_f P_{\mathcal{L}}(c|f,d)df = \frac{\Lambda(c)}{|Y|}\int_f df$ can be easily adapted to show that $\int_{f_{d_X}} P_{\mathcal{L}}(c|f,d)df_{d_X} = \frac{\Lambda(c)}{|Y|}\int_{f_{d_X}} df_{d_X}$. Hence

$$\begin{aligned}
\int_f P_{\mathcal{L}}(c|f,m)df &= \textstyle\sum_d \int_{f_{d_X}, f_{\sim d_X}} P_{\mathcal{L}}(d|f_{d_X},m)P_{\mathcal{L}}(c|f_{\sim d_X},d)df_{\sim d_X}df_{d_X} \\
&= \textstyle\sum_d \int_{f_{d_X}} P_{\mathcal{L}}(d|f_{d_X},m)[\int_{f_{\sim d_X}} P_{\mathcal{L}}(c|f_{\sim d_X},d)df_{\sim d_X}]df_{d_X} \\
&= \textstyle\sum_d \int_{f_{d_X}} P_{\mathcal{L}}(d|f_{d_X},m)\frac{\Lambda(c)}{|Y|}\int_{f_{\sim d_X}} df_{\sim d_X}df_{d_X} \\
&= \frac{\Lambda(c)}{|Y|}\textstyle\sum_d \int_{f_{d_X}, f_{\sim d_X}} P_{\mathcal{L}}(d|f,m)df_{\sim d_X}df_{d_X} \\
&= \frac{\Lambda(c)}{|Y|}\textstyle\sum_d \int_f P_{\mathcal{L}}(d|f,m)df \\
&= \frac{\Lambda(c)}{|Y|}\int_f \textstyle\sum_d P_{\mathcal{L}}(d|f,m)df \\
&= \frac{\Lambda(c)}{|Y|}\int_f df
\end{aligned}$$

■

The above NFL theorem actually hold under more general circumstances. A careful analysis of the proof of Theorem 11 will convince us that Theorem 11 holds as long as the data generation process guarantees that $P_{\mathcal{L}}(d|f,m)$ depends only on f_{d_X} and $P_{\mathcal{L}}(c|f,d)$ only depends on $f_{\sim d_X}$ for any f and d.

Wolpert also showed a similar NFL theorem holds for $P(c|d)$, the quantity of interest in Bayesian analysis. To be precise, $P(c|d) = \frac{\Lambda(c)}{|Y|}$, if $P(f)$ is uniform, $P_{\mathcal{L}}(d|f,m)$ depends only on f_{d_X} and $P_{\mathcal{L}}(c|f,d)$ only depends on $f_{\sim d_X}$ for any f and d. This can be seen by observing that $P(c|d) = \int_f P(f,c|d)df = \int_f P(f|d)P(c|d,f)df = \int_f \frac{P(d|f)P(f)}{P(d)}P(c|d,f)df = \int_f P(d|f)P(c|d,f)df/[P(d)\int_f df]$. Using the same decomposition of f in the proof above, the denominator can be evaluated to $\frac{\Lambda(c)}{|Y|}P(d)\int_f df$. Thus $P(c|d) = \frac{\Lambda(c)}{|Y|}$.

2.4.2 NFL for Expected Risk and Convergence Rate on Finite Samples

We briefly discuss two more NFL theorems for binary classification below.

A learning algorithm $\mathcal{L}$ is a map from training examples $(X_1, Y_1), \ldots, (X_n, Y_n)$ to a function $\mathcal{L}(X_1, Y_1, \ldots, X_n, Y_n) : X \to Y$. Let $\mathrm{R}_{\mathcal{L}}(n) = \mathrm{E}(L_{01}(X, Y, \mathcal{L}(X_1, Y_1, \ldots, X_n, Y_n))$. Note that the expectation is taken with respect to $X_1, Y_1, \ldots, X_n, Y_n, X, Y$.

It is known that there are many learning algorithms such that $\lim_{n\to\infty} \mathrm{R}_{\mathcal{L}}(n) = \mathrm{R}_{01}$ for any distribution P on $X \times Y$, where R_{01} is the Bayes risk. Such algorithm is said to be Bayes risk consistent.

The first result says that any generalization algorithm can have arbitrarily bad performance on a finite sample.

Theorem 12. *(Devroye, 1982) Let $X = \mathbf{R}^d$ and $Y = \{0, 1\}$. For any learning algorithm $\mathcal{L}$, for any n, for any $\epsilon > 0$ there exists a distribution P on $X \times Y$ such that $\mathrm{R}_{01} = 0$ but $\mathrm{R}_{\mathcal{L}}(n) \geq \frac{1}{2} - \epsilon$.*

The second result says that any Bayes risk consistent algorithm can have arbitrarily slow convergence rate.

Theorem 13. *(Devroye, 1982) Let $X = \mathbf{R}^d$ and $Y = \{0, 1\}$. For any learning algorithm $\mathcal{L}$, for any $\mathrm{R}^* \in [0, \frac{1}{2}]$, for any positive sequence a_n converging to 0, there exists a distribution P on $X \times Y$ such that $\mathrm{R}_{01} = \mathrm{R}^*$ and for infinitely many n, $\mathrm{R}_{\mathcal{L}}(n) \geq \min\{\mathrm{R}^* + a_n, \frac{1}{2}\}$.*

Thus for binary classification, only distribution-dependent results can be obtained for good performance guarantees on finite sample and fast convergence rate to Bayes risk.

2.4.3 Implications of NFL Theorems

We have seen NFL theorems demonstrating that in various cases, without sufficient prior knowledge, no learning algorithm can have guaranteed better performance than others.

But the the implication on the limitations of machine learning should not be taken too pessimistically. The scenarios considered in the NFL theorems do not cover all possible learning problems. For example, Wolpert's NFL theorems are for finite X and Y. In addition, Wolpert also demonstrated that under certain scenarios, there are cases for which one learning algorithm can have better performance compared to another (Wolpert, 1996b).

While the NFL theorems stress the necessity of prior knowledge for good performance guarantee, and thus the importance of designing learning algorithms by starting from realistic assumptions, it does not imply that useful knowledge cannot be obtained. Consider how scientists discover laws of nature. Initially there is no prior knowledge known to the scientists, but with the combination of logical reasoning and scientific experiments, no one can now deny that science has provided significant understanding of the true laws of nature. While in most cases there is no way to prove that a hypothesis is the true law, logical reasoning and experiments help us to reinforce certain beliefs and reject some others, and we learn useful knowledge which may serve as prior knowledge.

2.5 Looking ahead

General models and methods have been developed for humans to explore when constructing systems which can automate learning. But heavy human intervention is required in specifying the goals, representation and learning mechanism. A higher level of automated learning is to construct machines that can learn how to learn, that

is, systems that have certain capability of discovering useful representations, choosing suitable hypothesis spaces, experimenting with estimation methods and then performing automatic prediction. Ideally, humans should only provide feedbacks on whether the systems is doing the right thing, rather than specifying all the details.

There are still many problems to solve to reach such a level of automation. The machines need to be equipped with mechanisms to recognize various raw inputs and represent them in a suitable way. Currently, it is difficult for machines to learn the semantics of new forms of raw data. For example, while it is relatively easy to predefine some representation to perform symbolic integration, and a machine can solve an integration problem which humans specify in its internal format, it is difficult to ask the machine to solve the same problems if humans use a different system of notations. Leibniz's idea of universal representation of ideas is attractive here, but it should be noted that even such a universal representation can be found, and humans can use it as a standard method to communicate with the learning systems, it is unlikely to be useful in practice. We human beings often find certain representations are often superior than some others for given problems, and suitable representations often make problem solving much easier. Similar observations are abound in computer science. In addition, expressiveness often comes with the price of being computationally intractable.

Current programming languages do not facilitate programming such systems. They are limited in their abstraction power. We often need to code the same ideas in slightly different forms, such as writing many slightly different codes to perform cross-validation, or performing training and testing. While true learning generally requires self-correction, it is difficult for a program to correct itself once errors or bugs are detected.

We conclude this chapter by remarking that this short discussion is necessarily incomplete, but we hope the discussions have outlined a framework to systematically interpret, develop, and analyze machine learning algorithms.

Chapter 3

Log-Linearity and Markov Property

Log-linearity and Markov property are two important properties behind parametric statistical models having efficient learning and inference. The exponential family is a class of very expressive log-linear model for which domain knowledge can be easily integrated. Popular examples of unconditional exponential families include Naive Bayes classifiers and Markov chains, and popular examples of conditional exponential families include logistic regression, maximum entropy classifiers, and log-linear conditional random fields. Markov property in graphs leads to factorized models (the Bayesian networks and the Markov random fields), for which efficient inference algorithms ensue as a consequence of the Markov property. Our work on sparse models fall within the framework of log-linear conditional random fields.

This chapter presents principles leading the exponential family and Markov random fields, and the basic principles of inference and learning. Most results for log-linear MRFs/CRFs can be derived from the fact that they are both maximum entropy (MaxEnt) models and Markov models. For example, the MaxEnt principle is what results in the linear form and learning by maximum likelihood estimation. The Markov assumption results in a factorization which facilitates the design of efficient inference algorithms.

Here is the outline of this chapter. Section 3.1 introduces the concept of unconditional and conditional exponential families, and their basic properties. Section 3.2 shows that conditional exponential families are exactly the MaxEnt models. Section 3.3 discusses the computational difficulty of making predictions for conditional exponential families in general. Section 3.4 gives a general account on the theory and algorithms for maximum likelihood estimation, and describes the application of the method for the exponential forms. Section 3.5 discusses MRFs and CRFs.

3.1 Exponential Families

We shall mainly describe the results for discrete distributions, but it is often easy to see that similar results hold for continuous distributions.

3.1.1 The Exponential Form

Exponential families (Fisher, 1934; Darmois, 1935; Pitman, 567-579; Koopman, 1936) play a significant role in the study of probability distributions. It unifies many apparently very different distributions, such as Gaussian distributions, exponential distributions, binomial distributions, into a common form that allows various properties to be stated and derived in simple terms.

The density functions of exponential families take a simple exponential form. Let $f(X|\theta)$ denote a pdf parametrized by θ, where θ is in a subset Θ of $\mathbf{R}^d$. Then $\{f(X|\theta) : \theta \in \Theta\}$ is called an *exponential family* if there exist functions η, T, A, B such that

$$f(x|\theta) = e^{\eta(\theta)\cdot T(x)+A(x)-B(\theta)} \tag{3.1.1}$$

η is a vector-valued function of θ, and it is called the *natural parameter* of the exponential family. Unless otherwise stated, we shall now assume $\eta(\theta) = \theta$.

While the density function involves T, A, B, it suffices to specify T and A to determine the exponential family. This is because from Eq. 3.1.1, it follows that $e^{B(\theta)} = \sum_x e^{\theta \cdot T(x) + A(x)}$. $Z(\theta) = e^{B(\theta)}$ is called the *partition function*, and $B(\theta)$ is called the log partition function.

To evaluate the density function, the partition function need to be determined first. Generally there is no simple closed form expression for the partition function, and some clever algorithms need to be designed to evaluate the summation efficiently.

A useful property of the partition function is the following

$$\boldsymbol{\nabla} \ln Z(\theta) \quad = \quad \mathrm{E}(T(X)|\theta) \tag{3.1.2}$$

This can be seen by noting that

$$\boldsymbol{\nabla} Z(\theta) = \boldsymbol{\nabla} e^{B(\theta)} = \textstyle\sum_x T(x) e^{\theta \cdot T(x) + A(x)} = Z(\theta) \sum_x T(x) e^{\theta \cdot T(x) + A(x) - B(\theta)}$$

Thus it follows that the gradient of the log-likelihood is

$$\boldsymbol{\nabla} \ln f(x|\theta) = \mathrm{E}(T(X)|\theta) - T(x) \tag{3.1.3}$$

The exponential families has an interesting characterization: For a distribution family such that the domain of X does not depend on θ, it has a sufficient statistic with bounded dimension iff it is an exponential family (Koopman, 1936). In addition, it is easy to show that $T(x)$ is a sufficient statistic.

A.*Markov models with states and observations.* An important class of exponential families is the Markov models.

Let S be a set of states, and O be a set of observations. A Markov model generates a sequence of state-observation pairs as follows:

(a) Sample a state s_1 from a fixed state distribution P_S, and then sample an obser-

vation o_1 from a conditional distribution $P_{O|S}(\cdot|s_1)$.

(b) From $t = 2$ onwards, first sample a state s_t from a fixed transition distribution $P_{S|S}(\cdot|s_{t-1})$, then sample an observation o_t from $P_{O|S}(\cdot|s_t)$.

A Markov process is generally used to generate finite sequences. Let $s_0 = \emptyset$, then the probability that a length T sequence $(\langle s_1, o_1\rangle, \ldots, \langle s_T, o_T\rangle)$ is generated is given by

$$P(\langle s_1, o_1\rangle, \ldots, \langle s_T, o_T\rangle) = \prod_{t=1}^{T} P(s_t|s_{t-1})P(o_t|s_t). \tag{3.1.4}$$

The above probability can be written in an exponential form:

$$P(\langle s_1, o_1\rangle, \ldots, \langle s_T, o_T\rangle) = \prod_{s,s'} e^{\#(s,s')\ln P(s'|s)} \prod_{s,o} e^{\#(s,o)\ln P(o|s)}. \tag{3.1.5}$$

where $\#(s, s')$ and $\#(s, o)$ denote the number of times transition from s to s' and outputting o in state s are observed respectively in the given sequence.

Markov models with states unobserved are called hidden Markov models (HMMs) and have been widely used in machine learning. See (Rabiner, 1989)

3.1.2 The Conditional Version

A family of parametric conditional distribution $\{f(y|x, \theta) : \theta \in \Theta\}$ is called a *conditional exponential family* if there exists functions T, A, B such that

$$f(y|x, \theta) \quad = \quad e^{\eta(\theta)\cdot T(x,y)+A(x,y)-B(x,\theta)} \tag{3.1.6}$$

Again, unless otherwise stated, we shall consider the case when $\eta(\theta) = \theta$.

Similar to the unconditional case, a conditional exponential family is specified by T and A. $Z(x, \theta) = e^{B(x,\theta)}$ is called the partition function, and $B(x, \theta)$ the log partition

function.

$$\nabla_\theta \ln Z(x,\theta) \;=\; \mathrm{E}(T(x,y)|x,\theta) \tag{3.1.7}$$

For each given x, $f(\cdot|x,\theta)$ is an exponential family, thus an exponential family can be viewed as a conditional one with only one possible x value. On the other hand, for a exponential family $\{g((x,y)|\theta) : \theta \in \Theta\}$, the associated family of conditional distributions $\{g(y|x,\theta) : \theta \in \Theta\}$ is a conditional exponential family. To be precise, if $g((x,y)|\theta) = e^{\eta^T(\theta)T(x,y)+A(x,y)-B(\theta)}$, then the associated conditional distribution of Y given X has pdf $g(y|x,\theta) = e^{\eta^T(\theta)T(x,y)+A(x,y)-B(x,\theta)}$, where $B(x,\theta) = \ln \sum_y e^{\eta^T(\theta)T(x,y)+A(x,y)}$.

We give several examples of conditional exponential distributions below.

Example 14 (Logistic regression)**.** Let X be a random vector in $\mathbf{R}^n$, and Y be a discrete random variable taking values $\{1,\ldots,K\}$. Logistic regression perform density estimation using conditional distributions of the form

$$P(y|x) = \begin{cases} e^{\beta_y^T x}/(1+\sum_{i=1}^{K-1} e^{\beta_i^T x}), & y < K; \\ 1/(1+\sum_{i=1}^{K-1} e^{\beta_i^T x}), & y = K. \end{cases} \tag{3.1.8}$$

Clearly the distribution used is of exponential form.

The particular form of the conditional distribution can be derived in a general framework for constructing linear decision boundaries. Assume a class y is associated with a function $\delta_y(x)$ called the discriminant function of class y, and suppose the prediction rule is $\arg\max_y \delta_y(x)$. To make sure the decision boundaries are linear, it suffices to make sure that for any two class i,j, the decision boundary for $\delta_i(x) > \delta_j(x)$ is a hyperplane. For example, if each $\delta_i(x)$ is linear, then the decision boundaries are linear. If we want each $\delta_i(x)$ to represent a probability value that x is in class i, then linear functions do not work. But we can require $\ln \delta_i(x) - \ln \delta_K(x) = \beta_i^T x$ for each

$i < K$. In this case, $\delta_i(x) = \delta_K(x)e^{\beta_i^T x}$. Using the constraint $\sum_{i=1}^K \delta_i(x) = 1$, we have $\delta_K(x) = 1/(1 + \sum_{i=1}^{K-1} e^{\beta_i^T x})$, and $\delta_i(x) = e^{\beta_y^T x}/(1 + \sum_{i=1}^{K-1} e^{\beta_i^T x})$ for $i < K$. □

Example 15 (Conditional random fields (Lafferty et al., 2001) for sequence labeling)**.** Let $\mathcal{X}$ be the set of observations, $\mathcal{Y}$ be the set of labels. Let $\mathbf{x} \in \mathcal{X}^*$ and $\mathbf{y} \in \mathcal{Y}^*$ denote an observation sequence and a label sequence respectively. The problem of sequence labeling involves predicting label sequences for given observation sequences. An example is to label a sequence of words with the POS (part-of-speech) tags. Let $|\cdot|$ be the function which maps a sequence to its length. Let $f_1, \ldots, f_m$, be real-valued functions of the form $f_i(\mathbf{x}, \mathbf{y}, t)$ (defined when $|\mathbf{x}| = |\mathbf{y}|$, and $1 \leq t \leq |\mathbf{x}|$), and $\lambda = (\lambda_1, \ldots, \lambda_m)$. A CRF induces the following probability distribution

$$P(\mathbf{y}|\mathbf{x}, \lambda) = \exp(\textstyle\sum_i \sum_t \lambda_i f_i(\mathbf{x}, \mathbf{y}, t)) / \sum_{\mathbf{y}'} \exp(\sum_i \sum_t \lambda_i f_i(\mathbf{x}, \mathbf{y}', t)) \quad (3.1.9)$$

This is a conditional exponential family with λ as the natural parameter, and $(\sum_t f_1(\mathbf{x}, \mathbf{y}, t), \ldots, \sum_t f_m(\mathbf{x}, \mathbf{y}, t))$ as the sufficient statistic. □

Example 16 (Distributions on trees)**.** Let $\mathcal{G}$ be a set of undirected connected graphs. Let G, T be random variables denoting a graph from $\mathcal{G}$, and a spanning tree of the graph respectively. Let $f_1, \ldots, f_m$ be real-valued functions of the form $f(G, e)$, where $G \in \mathcal{G}$, and e is an edge of G, $\lambda = (\lambda_1, \ldots, \lambda_m)$ be a real vector. A conditional exponential family of T given G with parameter λ is the following

$$P(T|G, \lambda) = \exp(\textstyle\sum_i \sum_{e \in T} \lambda_i f_i(G, e)) / \sum_{T'} \exp(\sum_i \sum_{e \in T'} \lambda_i f_i(G, e)) \quad (3.1.10)$$

We can similarly define conditional exponential families of directed trees on directed graphs, or distributions on other classes of subgraphs, such as forests, if needed. □

3.2 Maximum Entropy Modeling

The exponential families can be characterized as a class of maximum entropy models. Maximum entropy modeling is a framework for estimating probability distributions under constraints derived from prior knowledge or data. There are two key developments behind this framework. One is Shannon's formulation of entropy as a measure on the uncertainty on probability distributions (Shannon, 1948), and Jaynes's *Maximum Entropy Principle*, which states that the most noncommittal probabilistic models satisfying given constraints should be preferred (Jaynes, 1957a,b). The maximum entropy principle is a generalization of Laplace's Principle of insufficient reason, which states that a uniform distribution should be preferred unless there is a reason to think otherwise. We shall explain these two developments below.

3.2.1 Entropy as a Measure of Uncertainty

Shannon's formulation of the measure of uncertainty in a probability distribution is a very surprising and aesthetically pleasing one. He showed that there is a unique (up to a positive multiplicative constant) such measure that is consistent with some intuitive properties on such measure.

Theorem 17. *(Shannon, 1948) There exists a unique (up to a positive multiplicative constant) real function H of discrete probability distributions satisfying the following properties:*

(a) (Continuity) For any discrete probability distribution p, H is nonnegative and continuous at p. [1]

[1] To make the notion of continuity precise, we need to define a metric space for discrete distributions. The can be done as follows. The set of discrete distributions can be defined as the set D of non-negative sequences having finitely non-zero elements summing to 1. For any $p_1, p_2 \in D$, let $\ell_1(p_1, p_2) \stackrel{\text{def}}{=}$

(b) (Monotonicity) For any positive integer n, $A(n) = H(\frac{1}{n}, \ldots, \frac{1}{n})$ is monotonically increasing in n.

(c) (The composition law) H satisfies

$$H(p_{11}, \ldots, p_{1n_1}, \ldots, p_{k1}, \ldots, p_{kn_k}) = H(w_1, \ldots, w_k) + \sum_{i=1}^{k} w_i H(\tfrac{p_{i1}}{w_i}, \ldots, \tfrac{p_{in_i}}{w_i}),$$

where $w_i = \sum_{j=1}^{n_i} p_{ij}$.

In fact, $H(p_1, \ldots, p_k) = -C \sum_i p_i \log p_i$.

Proof. By the composition law, we have

$$H(\frac{1}{mn}, \ldots, \frac{1}{mn}) = H(\frac{1}{n}, \ldots, \frac{1}{n}) + \sum_{i=1}^{n} \tfrac{m}{nm} H(\tfrac{1}{m}, \ldots, \tfrac{1}{m}),$$

that is, $A(mn) = A(m) + A(n)$, and thus $A(n^m) = mA(n)$.

For any positive integers n, k, we have $2^{\lfloor k \log n \rfloor} \leq n^k \leq 2^{\lceil k \log n \rceil}$, thus $A(2^{\lfloor k \log n \rfloor}) \leq A(n^k) \leq A(2^{\lceil k \log n \rceil})$, $\frac{\lfloor k \log n \rfloor}{k} A(2) \leq A(n) \leq \frac{\lceil k \log n \rceil}{k} A(2)$. Let $k \to \infty$, then we have $A(n) = A(2) \log n = C \log n$.

Let $n_1, \ldots, n_k$ be positive integers, and $n = n_1 + \ldots + n_k$, then by the composition law, $H(\frac{1}{n}, \ldots, \frac{1}{n}) = H(\frac{n_1}{n}, \ldots, \frac{n_k}{n}) + \sum_{i=1}^{k} \frac{n_i}{n} H(\frac{1}{n_i}, \ldots, \frac{1}{n_i})$. Thus $H(\frac{n_1}{n}, \ldots, \frac{n_k}{n}) = A(n) - \sum_{i=1}^{k} \frac{n_i}{n} A(n_i) = C \log n - \sum_{i=1}^{k} \frac{n_i}{n} C \log n_i = -C \sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$.

Hence $H(p_1, \ldots, p_k) = -C \sum_i p_i \log p_i$ for all positive rational p_i's. By the continuity of H, this holds for all non-negative real p_i's ■

We show that the composition law can be varied to give other sensible uncertainty measures.

Proposition 18. *Under the same conditions as in Theorem 17, but with the composition law (c) replaced by $H(p_{11}, \ldots, p_{1n_1}, p_{21}, \ldots, p_{2n_2}, \ldots, p_{k1}, \ldots, p_{kn_k}) = H(w_1, \ldots, w_k) +$*

$\sum_{i=1}^{\infty} |p_1(i) - p_2(i)|$, then (D, ℓ_1) is a metric space for discrete distributions.

$\sum_{i=1}^{k} w_i^\alpha H(\frac{p_{i1}}{w_i}, \ldots, \frac{p_{in_i}}{w_i})$, *where* $w_i = \sum_{j=1}^{n_i} p_{ij}$. *Then for any* $\alpha > 1$, *there is a unique solution of the form* $H(p_1, \ldots, p_k) = C(1 - \sum_{i=1}^{k} p_i^\alpha)$.

Proof. Let $A(n) = H(\frac{1}{n}, \ldots, \frac{1}{n})$, then for all positive integers n, m, $A(nm) = A(n) + \frac{1}{n^{\alpha-1}} A(m)$ and $A(nm) = A(m) + \frac{1}{m^{\alpha-1}} A(n)$. Let $m = 2$ and $C = 2A(2)$, then $C > 0$, and from the above two equations we have $A(n) = C(1 - \frac{1}{n^{\alpha-1}})$, where $C = A(2)/(1 - \frac{1}{2^{\alpha-1}})$. Let $n_1, \ldots, n_k$ be positive integers which sum to n, then $H(\frac{1}{n}, \ldots, \frac{1}{n}) = H(\frac{n_1}{n}, \ldots, \frac{n_k}{n}) + \sum_{i=1}^{k} (\frac{n_i}{n})^\alpha H(\frac{1}{n_i}, \ldots, \frac{1}{n_i})$. Hence $H(\frac{n_1}{n}, \ldots, \frac{n_k}{n}) = C(1 - \frac{1}{n^{\alpha-1}}) - \sum_{i=1}^{k} (\frac{n_i}{n})^\alpha C(1 - \frac{1}{n_i^{\alpha-1}}) = C(1 - \sum_{i=1}^{k} (\frac{n_i}{n})^\alpha$. Thus $H(p_1, \ldots, p_k) = C(1 - \sum_{i=1}^{k} p_i^\alpha)$ for all positive rational numbers $p_1, \ldots, p_k$. By continuity, this holds for all nonnegative real $p_1, \ldots, p_k$. ■

An interesting particular case is when $C = 1$ and $\alpha = 2$, then the corresponding entropy function is the probability that two independent draws from a categorical distribution parametrized by $(p_1, \ldots, p_k)$ have distinct outcomes.

3.2.2 The Principle of Maximum Entropy

The MaxEnt principle states that *among all consistent distributions, prefer the one having maximum entropy.* Informally, the most noncommittal consistent model is preferred. A model chosen according to the MaxEnt principle will be called a MaxEnt model.

As a simple illustration, suppose a random variable Z takes only n possible outcomes, then any distribution on Z corresponds to a point in the n dimensional unit simplex, that is, a tuple $(p_1, \ldots, p_n)$ such that $p_1, \ldots, p_n \geq 0$ and $p_1 + \ldots + p_n = 1$. If no other constraints are imposed, then the MaxEnt model can be easily shown to be the uniform distribution $(\frac{1}{n}, \ldots, \frac{1}{n})$. If the constraint $\sum_{i=1}^{n} i p_i = c$ (c a constant between 1 and n) is imposed, then the MaxEnt model is given by $p_i = e^{\lambda i} / \sum_{j=1}^{n} e^{\lambda j}$, where λ

satisfies $(n-c)e^{(n+1)\lambda} - (n-c+1)e^{n\lambda} + ce^{n\lambda} - (c-1) = 0$. Note that when $c = \frac{n+1}{2}$, then $\lambda = 0$ and we obtain the uniform distribution again; Let $c \to 1$, then we obtain the distribution $(1, 0, \ldots, 0)$; Let $c \to n$, then we obtain the distribution $(0, \ldots, 0, 1)$.

For the case when we are interested in learning the conditional distribution of a random variable Y given another random variable X and the distribution of X, the MaxEnt principle then states that *among all consistent conditional distributions, prefer the one having maximum conditional entropy.* We consider several useful conditional exponential families and show that each corresponds to a class of MaxEnt models.

3.2.3 Conditional Exponential Families as MaxEnt Models

We now derive conditional exponential families as MaxEnt models. Our derivation is in a more general setting than that in (Berger et al., 1996).

We shall only consider the case (X, Y) can have only finitely many values. The possible values of Y may depend on X (as in Example 15), and the set of these values will be denoted by $Y(X)$. Let π denote the distribution of X, p denote a conditional distribution of Y given X, $f_1, \ldots, f_m$ denote real-valued functions of X and Y, and $(x_1, y_1), \ldots, (x_n, y_n)$ denote a sequence of observed (X, Y) pairs.

π and p determine a joint distribution $P(X, Y)$ of X and Y. The expected value of f_i with respect to $P(X, Y)$ is given by $\mathrm{E}(f_i) = \sum_x \pi(x) \sum_y p(y|x) f_i(x, y)$. The conditional entropy of Y given X with respect to $P(X, Y)$ is given by $H(Y|X) = -\sum_{x \in X, y \in Y} \pi(x) p(y|x) \ln p(y|x)$. $H(Y|X)$ will be written as $H(p)$ to emphasize that it is viewed as a function of p.

$$
\begin{aligned}
\text{maximize} \quad & H(p) \\
\text{subject to} \quad & \textstyle\sum_{y\in Y} p(y|x) = 1, \quad \forall x \in X \\
& p(y|x) \geq 0, \quad \forall x \in X, \forall y \in Y(x) \\
& \mathrm{E}(f_i) = c_i, \quad \forall i
\end{aligned}
\tag{3.2.1}
$$

where c_i's are constants, which are generally derived from the observation and output pairs or some prior knowledge. c_i is often chosen to be $\tilde{\mathrm{E}}(f_i) = \frac{1}{n}\sum_i f_i(x_i, y_i)$, which is the empirical mean of f_i. Assume *Slator's condition is satisfied*, that is, there exists a p in the set $\mathcal{D}$ of distributions satisfying all the constraints, such that p is in the relative interior of $\mathcal{D}$ and all the inequality constraints hold strictly for p. Using standard Lagrangian duality method (cf. (Boyd and Vandenberghe, 2004, Chap. 5)), it can be shown that the MaxEnt model must be of exponential form.

Theorem 19. *The solution of Problem 3.2.1 is given by* $p(y|x) = \frac{e^{\sum_i \lambda_i f_i(x,y)}}{Z_x(\lambda)}$, *where* $\lambda = (\lambda_1, \ldots, \lambda_m)$, $Z_x(\lambda) = \sum_{y'} e^{\sum_i \lambda_i f_i(x,y')}$, *with* λ *being the unique minimizer of* $\sum_x \pi(x) \ln Z_x(\lambda) - \sum_i \lambda_i c_i$.

Proof. First consider the relaxed problem without the nonnegativeness constraint on $p(y|x)$. The Lagrangian of the relaxed problem is

$$
\mathcal{L}(p, \lambda, \mu) = -\textstyle\sum_{x,y} \pi(x) p(y|x) \ln p(y|x) + \sum_i \lambda_i (\mathrm{E}(f_i) - c_i) + \sum_x \mu_x (\sum_y p(y|x) - 1)
$$

where λ is the vector consisting of all λ_i's, and μ is the vector consisting of all μ_x's.

We now solve for the dual function $g(\lambda,\mu) \stackrel{\text{def}}{=} \sup_p \mathcal{L}(p,\lambda,\mu)$. We have

$$\frac{\partial \mathcal{L}}{\partial p(y|x)} = -\pi(x)(\ln p(y|x)+1) + \pi(x)\textstyle\sum_i \lambda_i f_i(x,y) + \mu_x = 0$$
$$\Rightarrow \quad p(y|x) = e^{\frac{\mu_x - \pi(x)}{\pi(x)}} e^{\sum_i \lambda_i f_i(x,y)}$$

The above p, which will be denoted by $p_{\lambda,\mu}$, is the only solution for $\boldsymbol{\nabla}_p \mathcal{L} = 0$ in the domain of $\mathcal{L}(\cdot,\lambda,\mu)$, and since $\mathcal{L}(p,\lambda,\mu)$ is strictly $\cap$-convex in p, $p_{\lambda,\mu}$ must be the unique maximizer of $\mathcal{L}(\cdot,\lambda,\mu)$.

Clearly Slator's condition is satisfied for the relaxed problem and the constraints are convex, strong duality holds as a consequence of Slator's theorem. That is, the maximum of the primal problem is $\min_{\lambda,\mu} g(\lambda,\mu)$, and if (λ,μ) is a minimizer of g, then $p_{\lambda,\mu}$ is a maximizer of the relaxed problem.

To simplify $\min_{\lambda,\mu} g(\lambda,\mu)$, note that strong duality also implies that if (λ,μ) minimizes g, then $p_{\lambda,\mu}$ is primal feasible, that is, it satisfies all primal constraints. Consider the primal constraint $\sum_{y'} p(y'|x) = 1$, then it follows

$$e^{-\frac{\mu_x - \pi(x)}{\pi(x)}} = Z_x(\lambda) \stackrel{\text{def}}{=} \textstyle\sum_{y'} e^{\sum_i \lambda_i f_i(x,y')}$$
$$\Rightarrow \quad p_{\lambda,\mu}(y|x) = \frac{e^{\sum_i \lambda_i f_i(x,y)}}{Z_x(\lambda)}$$

Substituting this expression back into $g(\lambda,\mu) = L(p_{\lambda,\mu},\lambda,\mu)$, then $g(\lambda,\mu) = h(\lambda) \stackrel{\text{def}}{=} \sum_x p(x) \ln Z_x(\lambda) - \sum_i \lambda_i c_i$. Hence the maximum of the primal is $\min_\lambda h(\lambda)$.

Note that h has a unique minimizer due to its strict $\cup$-convexity.

The proof is completed by noting that the above solution for the relaxed problem satisfies the nonnegativeness constraints, thus it is also the the solution for the original problem. ■

An alternative approach in the above is to consider the primal constraint $\mathrm{E}(f_i) = c_i$

in our attempt to derive a simpler expression for $\min_{\lambda,\mu} g(\lambda,\mu)$ and $p_{\lambda,\mu}$, but these constraints do not help us to reduce the number of variables. However, these constraints are implicitly used, as they are equivalent to λ being the minimizer of $h(\lambda)$. Thus we can either characterize the parameter of the conditional exponential distribution as the solution of an optimization problem or the solution of a system of equations. These two views lead to two types of numerical algorithms for the MaxEnt parameter.

In practice, $\pi(x)$ may not be given and the empirical distribution $\tilde{\pi}(x)$ is used for $\pi(x)$. In this case, there is an interesting connection between MaxEnt principle and ML estimation, which can be obtained as an immediate corollary of the above result.

Corollary 20. *If $\pi(x) = \tilde{\pi}(x)$, then the solution of Problem 3.2.1 is given by the distribution $p(y|x) = \frac{e^{\sum_i \lambda_i f_i(x,y)}}{Z_x(\lambda)}$ with λ chosen to maximize $\ln \prod_i p(y_i|x_i) + \sum_j \lambda_j (c_j - \tilde{\mathrm{E}}(f_j))$. In particular, when $c_i = \tilde{\mathrm{E}}(f_i)$ for each f_i, then λ is chosen to maximize the log-likelihood $L(\{(x_i,y_i)\}|\lambda) = \ln \prod_i p(y_i|x_i)$.*

Thus for this case, the MaxEnt principle corresponds to using conditional exponential distributions as the model, and maximum likelihood estimation as the estimation method.

The following corollary states the equivalence of MaxEnt models and conditional exponential distributions.

Corollary 21. *Let $p_\lambda^{exp}(y|x) = \frac{e^{\sum_i \lambda_i f_i(x,y)}}{Z_x(\lambda)}$. Let $c = (c_1,\ldots,c_m)$, and p_c^{me} be the MaxEnt model for Problem 3.2.1. Let $\mathcal{C}$ be the set of $(c_1,\ldots,c_m)$ such that Problem 3.2.1 has a solution. Then*

(a) For every $c \in \mathcal{C}$, there exists a unique $\lambda \in \mathbf{R}^m$ such that $p_c^{me} = p_\lambda^{exp}$.

(b) For every $\lambda \in \mathbf{R}^m$, there exists a unique $c \in \mathcal{C}$ such that $p_\lambda^{exp} = p_c^{me}$.

Proof. (a) follows from Theorem 19 directly. For (b), given λ, the corresponding c is given by $c_i = \sum_x \pi(x) \sum_y p_\lambda^{exp}(y|x) f_i(x,y)$. ■

When Y is a discrete random variable, then we can use the MaxEnt model to classify an instance x into class $\arg\max_y p(y|x)$. This is the MaxEnt classifier. The difference between this MaxEnt distribution and the distribution used for logistic regression is that there is a reference class K in logistic regression, while every class is treated equally in MaxEnt. Let $f_{i,y}(x, y') = x(i)\mathrm{I}(y = y')$, where $x(i)$ is the value of the i-th attribute of x, $1 \leq i \leq d$ and $1 \leq y < K$. The MaxEnt formulation for the distribution in logistic regression is

$$\begin{array}{lll} \text{maximize} & H(p) & \\ \text{subject to} & \sum_{y \in Y} p(y|x) = 1, & \forall x \in X \\ & p(y|x) \geq 0, & \forall x \in X, \forall y \in Y \\ & \mathrm{E}(f_{i,y}) = c_i, & \forall 1 \leq i \leq d, \forall 1 \leq y < K \end{array} \tag{3.2.2}$$

Remark 22. (a) If we use $H(X) = 1 - \sum_x p^2(x)$ as the measure of entropy, then the nonnegativeness constraints cannot be ignored now. A dual variable need to be introduced for each nonnegativeness constraint. The resulting dual problem involves many more variables and is much harder to solve, while using Shannon's entropy formula we can derive an elegant distribution parametrized by a small number of variables.

(b) The exponential family derived above does not have a term equivalent to the term $A(x, y)$ in the definition for conditional exponential family. To introduce such a term, it suffices to replace $H(p)$ in Problem 3.2.1 by $H(p) + \mathrm{E}(A)$. Such modification on the objective can be interpreted as imposing a constraint $\mathrm{E}(A) = a$ but with its corresponding dual variable being forced to 1.

If $A(x, y) = \ln q(x, y)$ for some distribution q of X and Y, then $-H(p) - \mathrm{E}(A)$ is $KL(p||q) = \sum_x \pi(x) \sum_y p(y|x) \ln \frac{p(y|x)}{q(y|x)}$, which is the conditional entropy (or KL-divergence) of p relative to q. Note that if q is fixed, then $KL(p||q)$ has minimum value 0 when $p = q$. Thus in this case, the objective encodes the preference for a

model which is closest to a given distribution q under given constraints. Maximizing $H(p)$ corresponds to the special case of minimizing $KL(p||q)$ with q being the uniform distribution.

If $\pi(x) = \tilde{\pi}(x)$ and $c_i = \tilde{\mathrm{E}}(f_i)$, then the parameters are again estimated by maximizing the likelihood.

A.*Example: Maximum Entropy Markov Models.* We consider estimating a conditional analogue of the generative Markov model. Let $X = (O_1, \ldots, O_T)$ be a random observation sequence, and $Y = (S_1, \ldots, S_T)$ the corresponding label sequence, where T may be different for different observed (X, Y) pairs. The conditional analogue considered is a distribution of the form

$$P(Y|X) = P(S_1, \ldots, S_T|X) = \prod_{t=1}^{T} P(S_t|S_{t-1}, X)$$

Given n observed (X, Y) pairs $(x_1, y_1), \ldots, (x_n, y_n)$, a conditional distribution of Y given X can be obtained by first estimating for each s_o, the conditional probability $P(s|s_o, x)$ of current state s given previous state s_o and observation sequence x, then combining them using Eq. 1. Let $f_1, \ldots, f_m$ be real-valued functions of X, t and S, where t is a position variable.

Consider estimating $P(s|s_o, x)$ for a fixed s_o. For a sequence y, let $y(t)$ denote the t-th element of y. For estimating $P(s|s_o, x)$, an observation variable Z is a pair (x_i, t) such that $y_i(t-1) = s_o$. Let $\pi(Z)$ denote the distribution of Z, and $p(s|Z)$ be the conditional distribution to be estimated. $\pi(Z)$ and $p(s|Z)$ gives a joint distribution $P(Z, S)$, and the conditional entropy is $H(p) = -\sum_z \pi(z) \sum_s p(s|z) \ln p(s|z)$. A

MaxEnt formulation is as follows.

$$\begin{array}{lll} \text{maximize} & H(p) & \\ \text{subject to} & \sum_{s\in S} p(s|z) = 1, & \forall z \in Z \\ & p(s|z) \geq 0, & \forall z \in Z, \forall s \in S \\ & \mathrm{E}(f_i) = c_i, & \forall i \end{array} \tag{3.2.3}$$

Applying Theorem 19 and Corollary 20, we can show the following.

(a) The solution for Problem 3.2.3 is of the form $p(s|x,t) = \frac{e^{\sum_i \lambda_i f_i(x,t,s)}}{Z_{x,t}(\lambda)}$, where $\lambda = (\lambda_1, \ldots, \lambda_m)$, $Z_{x,t}(\lambda) = \sum_{s'} e^{\sum_i \lambda_i f_i(x,t,s')}$, with λ being the unique minimizer of $\sum_{x,t} \pi(x,t) \ln Z_x(\lambda) - \sum_i \lambda_i c_i$.

(b) When $\pi(x,t) = \tilde{\pi}(x,t)$, and $c_i = \tilde{\mathrm{E}}(f_i)$ for each f_i, then λ is (a) is equivalently the unique maximizer of the log-likelihood $L(\{\langle (x_i,t), y_i(t)\rangle : y_i(t-1) = s_o\}|\lambda) = \ln \prod_{y_i(t)=s_o} p(y_i(t)|x_i,t)$.

By constraining each $f_i(x,t,s)$ to depend on $x(t)$ and s only, the above model is then reduced to the Maximum Entropy Markov Models (MEMM) (McCallum et al., 2000).

B.*Example: Conditional random fields.* Now we show that the CRFs in Example 15 can be derived as a MaxEnt model. As in previous section, X and Y denote a random observation and a random label sequence respectively. Let $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$ be n observed (X,Y) pairs, $f_1, \ldots, f_m$ be real-valued functions of X, Y, and t, where t is a position variable. $Y(X)$ is the set of label sequences of the same length as X.

Let $\pi(X)$ denote the distribution of X, and $p(Y|X)$ be the conditional distribution to be estimated. $\pi(X)$ and $p(Y|X)$ gives a joint distribution $P(X,Y)$, and the conditional entropy is $H(p) = -\sum_{\mathbf{x}} \pi(x) \sum_{\mathbf{y}\in Y(x)} p(\mathbf{y}|\mathbf{x}) \ln p(\mathbf{y}|\mathbf{x})$. Let $\bar{f}_i(\mathbf{x},\mathbf{y}) \stackrel{\text{def}}{=}$

$\sum_{t=1}^{|\mathbf{x}|} f_i(\mathbf{x}, \mathbf{y}, t)$. A MaxEnt formulation is as follows.

$$\begin{array}{lll} \text{maximize} & H(p) & \\ \text{subject to} & \sum_{\mathbf{y} \in Y(\mathbf{x})} p(\mathbf{y}|\mathbf{x}) = 1, & \forall \mathbf{x} \in X \\ & p(\mathbf{y}|\mathbf{x}) \geq 0, & \forall \mathbf{x} \in X, \forall \mathbf{y} \in Y(\mathbf{x}) \\ & \mathrm{E}(\bar{f}_i) = c_i, & \forall i \end{array} \tag{3.2.4}$$

Now Theorem 19 can be applied directly to yield the CRFs in Example 15.

C.*Example: Conditional distributions on trees.* As another example, we point out that the solution to the following problem is of the form in Example 16.

$$\begin{array}{lll} \text{maximize} & H(p) & \\ \text{subject to} & \sum_{t \in \mathcal{T}(G)} p(T|G) = 1, & \forall G \in \mathcal{G} \\ & p(T|G) \geq 0, & \forall G \in \mathcal{G}, \forall T \in \mathcal{T}(G) \\ & \mathrm{E}(\bar{f}_i) = c_i, & \forall i \end{array} \tag{3.2.5}$$

where $\bar{f}_i(G, T) \stackrel{\text{def}}{=} \sum_{e \in T \in \mathcal{T}(G)} f_i(G, e)$.

D.*Inequality Constraints.* So far only linear equality constraints have been discussed. When linear inequality constraints are considered, we will see an interesting connection between the MaxEnt model and regularization. We return to Problem 3.2.1, and consider the following variant, which replaces the equality constraint on expectation

by inequality constraints.

$$\begin{array}{lll} \text{maximize} & H(p) & \\ \text{subject to} & \sum_{y\in Y} p(y|x) = 1, & \forall x \in X \\ & p(y|x) \geq 0, & \forall x \in X, \forall y \in Y \\ & \mathrm{E}(f_i) \geq l_i, & \forall i \\ & \mathrm{E}(f_i) \leq u_i, & \forall i \end{array} \tag{3.2.6}$$

Proposition 23. *(a) The solution for Problem 3.2.6 is* $p(y|x) = e^{\sum_i \lambda_i f_i(x,y)}/Z_x(\lambda)$, *where* $Z_x(\lambda) = \sum_{y'} e^{\sum_i \lambda_i f_i(x,y')}$, *and* λ *minimizes* $h(\lambda) = \sum_x \pi(x) \ln Z_x(\lambda) - \sum_i \min(\lambda_i l_i, \lambda_i u_i)$.

(b) When $\pi(x) = \tilde{\pi}(x)$, $l_i = \tilde{\mathrm{E}}(f_i) - \frac{w_i}{n}$, $u_i = \tilde{\mathrm{E}}(f_i) + \frac{w_i}{n}$ *(each* w_i *is a positive constant),* λ *in (a) is equivalently the maximizer of* $\sum_i \ln p(y_i|x_i) - \sum_i w_i|\lambda_i|$.

Proof. We sketch the main steps in the derivation of the solution.

The Lagrangian for the relaxed problem without the nonnegativeness constraints is

$$\mathcal{L}(p, \hat{\lambda}, \breve{\lambda}, \mu) = -\sum_{x,y} \pi(x)p(y|x) \ln p(y|x) + \sum_i \breve{\lambda}_i(\mathrm{E}(f_i) - l_i) + \sum_i \hat{\lambda}_i(u_i - \mathrm{E}(f_i)) + \sum_x \mu_x(\sum_y p(y|x) - 1)$$

where $\hat{\lambda}_i \geq 0, \breve{\lambda}_i \geq 0$ for all i.

First setting $\boldsymbol{\nabla}_p \mathcal{L} = 0$ to find the dual function, and then applying strong duality and the constraints $\sum_y p(y|x) = 1$ to eliminate μ in the dual function, the solution to the relaxed problem is then found out to be $p(y|x) = e^{\sum_i (\breve{\lambda}_i - \hat{\lambda}_i) f_i(x,y)}/Z_x(\hat{\lambda}, \breve{\lambda})$, where $Z_x(\hat{\lambda}, \breve{\lambda}) = \sum_{y'} e^{\sum_i (\breve{\lambda}_i - \hat{\lambda}_i) f_i(x,y')}$, and $\hat{\lambda}, \breve{\lambda}$ are nonnegative vectors minimizing $g(\hat{\lambda}, \breve{\lambda}) = \sum_x \pi(x) \ln Z_x(\hat{\lambda}, \breve{\lambda}) - \sum_i \breve{\lambda}_i l_i + \sum_i \hat{\lambda}_i u_i$.

Let $\lambda = \breve{\lambda} - \hat{\lambda}$, then $p(y|x)$ as a function of λ, and $g(\hat{\lambda}, \breve{\lambda}) = \sum_x \pi(x) \ln Z_x(\lambda) - \sum_i \lambda_i l_i + \sum_i \hat{\lambda}_i(u_i - l_i)$, where λ is unconstrained, and $\hat{\lambda}_i \geq \max(0, -\lambda_i)$. For fixed λ, $g(\hat{\lambda}, \breve{\lambda})$ has minimum $h(\lambda) = \sum_x \pi(x) \ln Z_x(\lambda) - \sum_i \min(\lambda_i l_i, \lambda_i u_i)$. Hence the λ

component in the minimizer of g is the same as the minimizer of h.

Now Proposition 23 follows easily. ■

3.3 Prediction

As in general, predictions for exponential families can be difficult even for simple loss functions due to the rich structures that X and Y can assume and the rich dependencies that can be present for X and Y. As an illustration, consider prediction using 0/1 loss, then we need to compute a most likely Y for any given X. We construct an exponential family such that if it is possible to do so in polynomial time, then SAT can be solved in polynomial time. Let X be a random set of clauses with 3 literals, and for any x, $Y(x)$ denotes a random assignment to the corresponding set of variables. Let x denote a set of clauses $\{c_1, \ldots, c_n\}$, y denote an assignment $(u_1, \ldots, u_m)$ to the corresponding set of variables, and $f(y|x, \lambda) \propto \exp(\lambda \sum_{i=1}^{n} \mathrm{I}(y \text{ satisfies } c_i))$. Choose $\lambda = 1$, then there is a satisfying assignment iff a most likely y is a satisfying assignment. The above reduction is clearly a polynomial-time reduction. Thus if we can find a most likely y for any x in polynomial time, then SAT can be solved in polynomial time.

In general, we can give a polynomial time reduction from an arbitrary NP problem to the problem of computing a most likely Y for a given X for a particular conditional exponential family. Let Σ be a finite alphabet, and $L \subseteq \Sigma^*$ be a language in NP. By definition, there exists a finite alphabet Ξ, a language $L' \subseteq \Sigma^* \times \Xi^*$ in P and a polynomial p such that $L = \{x : \exists (x, y) \in L' \text{ such that } |y| \leq p(|x|)\}$, where $|\cdot|$ is the length function. Let $X = \Sigma^*$, $Y(x) = \Xi^{p(|x|)}$, and $f(y|x, \lambda) \propto \exp(\lambda \mathrm{I}((x, y) \in L'))$. Choose $\lambda = 1$, then x is in L iff a most likely value y for x satisfies $(x, y) \in L'$. Thus we have a polynomial time reduction from the membership problem of L to the problem of computing a most likely value for any x for the conditional exponential family $f(y|x, \lambda)$.

Many other problems for conditional exponential families are computationally hard. In particular, even evaluating the pdf are generally difficult. This is often due to the difficulty of computing the partition function, which can require summation over exponentially many y's. A general intractability result on computing partition functions was established by Sorin Istrail's (Istrail, 2000).

Approximation methods are often applied in such cases. These include sampling methods (cf. (Andrieu et al., 2003)) and variational approximation methods (cf. (Jordan et al., 1999; Wainwright and Jordan, 2008)).

3.4 Learning

3.4.1 Maximum Likelihood Estimation

Given a family of distributions, the problem of learning is to find a member which is closest to the true distribution, using data generated by the true distribution. For parametric families, this is called parameter estimation. Thus a parameter estimation method is a mapping from observed data to a parameter, and will often be called an *estimator*. In practice, a parametric family is often *misspecified*, that is, it does not contain the true distribution.

For some special cases, the parameters have intuitive interpretations which allow them to be estimated from data using simple computation procedures. For example, the parameters in naive Bayes have simple frequentist interpretations which suggest estimating them using empirical frequencies. Another example is that the parameters μ and σ^2 in the normal distribution $f(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, are the expectation and variance of X respectively, thus we can estimate them using the empirical mean and variance.

For general cases, including the exponential families, there are no simple closed-form estimation formula, but there is a general criterion due to Fisher, which he proposed

and consolidated from 1912 to 1922, during which he gave three different justifications to the method ((Fisher, 1912, 1921, 1922); see (Aldrich, 1997) for a detailed account on the development of the principle). This is Fisher's *maximum likelihood* (*ML*) estimator, or *MLE*. We shall use MLE to denote ML estimation, ML estimate as well, and the meaning will be clear from context.

For a generative distribution $f(x|\theta)$, let $L(x_1, \ldots, x_n|\theta)$ be the joint probability that $x_1, \ldots, x_n$ are observed. L is regarded as a function of θ and called the *likelihood function*. Then the MLE is defined as $\theta^{ML} = \arg\max_\theta L(x_1, \ldots, x_n|\theta)$. If $x_1, \ldots, x_n$ are generated independently, then $L(x_1, \ldots, x_n|\theta) = \prod_{i=1}^n f(x_i|\theta)$. For a conditional distribution $f(y|x, \theta)$, the likelihood function $L(y_1, \ldots, y_n|x_1, \ldots, x_n, \theta)$ is defined as the joint probability of observing $y_1, \ldots, y_n$ given $x_1, \ldots, x_n$. It is often easier to work with the log-likelihood function $\ell(x_1, \ldots, x_n|\theta) \stackrel{\text{def}}{=} \ln L(x_1, \ldots, x_n|\theta)$, and $\ell(y_1, \ldots, y_n|x_1, \ldots, x_n, \theta) \stackrel{\text{def}}{=} \ln L(y_1, \ldots, y_n|x_1, \ldots, x_n, \theta)$.

An information theoretic interpretation of the MLE p from a family $\mathcal{F}$, is that p minimizes the KL-divergence $KL(\tilde{p}||p)$, where $\tilde{p}$ is the empirical distribution.

The Naive Bayes estimates are actually the MLEs. We have also given a justification of using log loss (thus ML) in Section 2.1 as well, where we argue that log loss can be used because the minimizer of $\mathrm{E}(\ln p(x, y))$ is the true distribution.

Many other important estimators maximize variants of likelihood. For example, a *maximum a posterior* (*MAP*) estimator maximizes a regularized likelihood instead of the likelihood. A pseudo-likelihood estimator replaces the true likelihood with an approximation (Besag, 1975). A composite likelihood estimator maximizes a weighted product of marginal or conditional probabilities (Lindsay, 1988).

MLE has been proved to possess natural properties that are desirable for estimators. Wald showed that MLE is consistent in the sense that it almost surely converges to the true parameter (Wald, 1949). Le Cam further showed that under weak regularity conditions, MLE is asymptotically normal (Le Cam, 1953). The misspecified case were

first considered by Berk (Berk, 1966, 1970) and Huber (Huber, 1967). Berk showed that the MLE still converges to a limit, and Huber shows under very general conditions that MLE converges to a limit and is asymptotically normal. White showed under more intuitive but more restrictive conditions MLE consistency and asymptotic normality hold (White, 1982). The results for the well-specified case can be viewed as a special case of the misspecified case. These results are summarized in the theorem below.

Theorem 24. *(White, 1982) Let $x_1, x_2, \ldots, x_n$ be i.i.d. observations drawn according to some distribution $g(X)$, $\{f(x|\theta) : \Theta\}$ be a parametric family with Θ being a subset of $\mathbf{R}^d$, and $l(x_1, \ldots, x_n|\theta) = \frac{1}{n}\sum_{i=1}^{n} \ln f(x_i|\theta)$ be the average log-likelihood function. Let*

$$A_n(\theta) = (\frac{\partial^2 l}{\partial\theta_i \partial\theta_j})_{ij},\ A(\theta) = (\mathrm{E}(\frac{\partial^2 \ln f}{\partial\theta_i \partial\theta_j}))_{ij},\ B_n(\theta) = (\frac{\partial l}{\partial\theta_i}\frac{\partial l}{\partial\theta_j})_{ij},\ B(\theta) = (\mathrm{E}(\frac{\partial \ln f}{\partial\theta_i}\frac{\partial \ln f}{\partial\theta_j})_{ij}$$

Assume $\mathrm{E}(\ln f(x|\theta))$ has a unique maximizer θ^*, and for all $x_1, \ldots, x_n$, l always has a (not necessarily unique) maximizer θ_n, then under some weak conditions [2], the following hold

(a) (Consistency) $\theta_n \xrightarrow{\text{a.s.}} \theta^*$, that is, $\theta_n \to \theta^*$ for almost every sequence (x_i).

(b) (Asymptotic normality) $\sqrt{n}(\theta_n - \theta^*) \xrightarrow{d} N(0, C(\theta^*))$, that is $\sqrt{n}(\theta_n - \theta^*)$ converges in distribution to $N(0, C(\theta^*))$, where $C(\theta) = A(\theta)^{-1}B(\theta)A(\theta)^{-1}$.

Moreover, let $C_n(\theta) = A_n(\theta)^{-1}B_n(\theta)A_n(\theta)^{-1}$, then $C_n(\theta_n) \xrightarrow{\text{a.s.}} C(\theta^*)$ element by element.

We point out that we can use the above results for unconditional distribution to obtain corresponding results for conditional distributions. The idea is that for a family of conditional distributions $\{f(y|x,\theta) : \theta \in \Theta\}$, we can consider the family $\{g_f((x,y)|\theta) : \theta \in \Theta\}$, where $g_f((x,y)|\theta) = \pi(x)f(y|x,\theta)$, then apply the results above to this family.

[2] See the cited paper for the list of conditions needed.

We mention two more properties of MLE. MLE is parametrization invariant, that is, if a family is parametrized by $\eta = s(\theta)$ and η^{ml} is an MLE given i.i.d. observations $x_1, \ldots, x_n$, then when the family is reparametrized using θ, the MLE θ^{ml} given $x_1, \ldots, x_n$ satisfies $\eta^{ml} = s(\theta^{ml})$. In addition, if a sufficient statistic t exists for the parameter θ, then the MLE is a function of t.

Remark 25. As discussed in Section 2.1, the parameters can also be chosen using the optimization approach, but note that in this case we are no longer estimating densities.

Remark 26. When the model is misspecified, systematic errors can occur. In this case, it may be useful to try to reduce systematic errors. For example, suppose in binary classification problem, if the probabilities are always biased towards + due to misspecification, then using the MAP classification rule may not be a good choice. Instead, it may be better to use a thresholded classification rule like $h_t(x) = \begin{cases} +, P(+|x) > t; \\ -, \text{otherwise.} \end{cases}$, where t is some threshold value that is learned from data.

3.4.2 MLE for the Exponential Forms

A.*Basic Equations.* Consider an exponential family with pdf $f(x|\theta) = e^{\theta^T T(x) + A(x) - B(\theta)}$. The log-likelihood function $\ell(x_1, \ldots, x_n|\theta)$ is $\cap$-convex, thus it has a unique maximizer, which is the solution to $\boldsymbol{\nabla}\ell = 0$, where $\boldsymbol{\nabla}\ell$ is given below.

$$\boldsymbol{\nabla}\ell(x_1, \ldots, x_n|\theta) = \textstyle\sum_{i=1}^n (T(x_i) - \mathrm{E}(T(x)|\theta)) \tag{3.4.1}$$

For the conditional exponential family, the analogue of the above equation is

$$\boldsymbol{\nabla}\ell((x_1, y_1), \ldots, (x_n, y_n)|\theta) = \textstyle\sum_{i=1}^n (T(x_i, y_i) - \mathrm{E}(T(x_i, y)|\theta)) \tag{3.4.2}$$

B.*Incomplete data.* Similar result can be derived for the case of incomplete data.

Let Z be a function of X, and let $X(z)$ denote the set of x's which are mapped to z, then the pdf of Z is $f(z|\theta) = \sum_{x \in X(z)} f(x|\theta)$. The gradient of log-likelihood function is given by

$$\nabla \ell(z_1, \ldots, z_n)|\theta) = \sum_{i=1}^{n} (\mathrm{E}(T(x)|z_i, \theta) - \mathrm{E}(T(x)|\theta)) \tag{3.4.3}$$

Eq. 3.4.1 can be seen as a special case of the above equation. However, the log-likelihood function is no longer $\cap$-convex, thus it may have several maximizers, and a solution to the above equation is not necessarily a global maximizer.

The conditional version of Eq. 3.4.4 is given by

$$\nabla \ell((x_1, z_1), \ldots, (x_n, z_n))|\theta) = \sum_{i=1}^{n} (\mathrm{E}(T(x_i, y)|z_i, \theta) - \mathrm{E}(T(x_i, y)|\theta)) \tag{3.4.4}$$

where Z is a function of Y.

C.*Factorizable models.* In practice, the random variables can have very complex structure, and the functions T and A, can be very complex and thus make it computationally intractable to compute the above expectations directly. But for many interesting cases, it is possible to design some models which can capture real-world information sufficiently well and yet computationally tractable. One of the most common method is to decompose T as a sum of factors which depend on local structures. Formally, consider a conditional exponential family with pdf $f(y|x, \theta) = e^{\theta \cdot T(x,y) + A(x,y) - B(x,\theta)}$. Suppose $T(x, y) = \sum_{\alpha \in I(x,y)} T(x, \alpha)$ for some set $I(x, y)$ and function $T(x, \alpha)$. $I(x, y)$ is called a structural index set, and f is said to be factorizable over I. To illustrate, the distribution on trees in Example 16 is factorizable over $I(G, T)$ chosen to be the edge set of T. To see this, note that the distribution corresponds to $T(G, T) = \sum_{e \in I(G,T)} T(G, e)$, $A(G, T) = 0$, where $T(G, e) = (f_1(G, e), \ldots, f_m(G, e))$.

For a factorizable model, the expectation of $T(x_i, y)$ can be decomposed.

$$\mathrm{E}(T(x_i, Y)|\theta) = \sum_{\alpha \in I(x_i)} \mu(\alpha|x_i) T(x_i, \alpha) \tag{3.4.5}$$

where $I(x_i) = \cup_y I(x_i, y)$, and $\mu(\alpha|x_i) = \sum_{\alpha \in I(x_i,y)} f(y|x_i, \theta)$ is called the marginal of the structural index α. Thus the problem of computing the expectation of $T(x_i, y)$ is reduced to the problem of computing the probability that $I(x_i, y)$ includes some structural index α. For the distributions on trees, this amounts to computing the probability that a tree contains an edge e.

The above discussion also applies for the case when the data is incomplete.

With suitably chosen structural indices, the marginals of the structural indices can be easy to compute. For example, see (Lafferty et al., 2001) and (Koo et al., 2007).

3.4.3 Algorithms for Computing Parameter Estimates

As in the case after Theorem 19, there are two different ways to characterize MLEs. One is to see them as solutions to systems of equations, and the other is to see them as solutions to an optimization problem. When they are seen as solutions to a system of equations, they can be found by using Newton-Raphson's method to solve the equations. When they are seen as solutions to a optimization problem, they can be found using typical optimization algorithms. These include various gradient based algorithms such as generalized iterative scaling (Darroch and Ratcliff, 1972), Quasi-Newton algorithms (such as the L-BFGS (Liu and Nocedal, 1989) algorithm, a memory-limited Quasi-Newton method), and stochastic gradient descent algorithms (Bottou, 2004).

When the data is incomplete, the Expectation-Maximization (EM) algorithm (Dempster et al., 1977) is often employed for maximizing the likelihood. Formally, let Z be a function of X, let $x_1, \ldots, x_n$ be i.i.d. observations generated using $f(X|\theta)$, and $z_1, \ldots, z_n$ be the corresponding Z values. In general, there are several values of X that correspond to the same Z value. To find a local maximizer θ for the joint probability $P(z_1, \ldots, z_n|\theta)$, the EM algorithm starts with an arbitrary initial weight θ_0, then iterates over the following two steps

- E step: $Q(\theta|\theta_m) = \sum_{x_1,\ldots,x_n} P(x_1, \ldots, x_n|z_1, \ldots, z_n, \theta_m) \ln P(x_1, \ldots, x_n|\theta)$, where

summation is over $x_1, \ldots, x_n$ consistent with $z_1, \ldots, z_n$.

- M step: $\theta_{m+1} = \arg\max_\theta Q(\theta|\theta_m)$.

The expectation step finds the expectation of the complete log-likelihood with respect to the distribution of $x_1, \ldots, x_n$ according to θ_m. The E step can be performed using sampling methods. The M step can be relaxed by requiring θ_{m+1} to satisfy $Q(\theta_{n+1}|\theta_n) \geq Q(\theta_n|\theta_n)$ only, and the algorithm is called a generalized EM (GEM) algorithm. This inequality is sufficient to guarantee that $P(z_1, \ldots, z_n|\theta_m)$ is non-decreasing as m increases.

3.5 Conditional Random Fields

Log-linear CRFs are an important subclass of conditional exponential families. They have been used with empirical successes in various sequence labeling problems, including labeling words in sentences with its type in named-entity recognition problems (Tjong Kim Sang and De Meulder, 2003), handwriting recognition problems (Taskar et al., 2003), deciding whether each DNA base in a DNA sequence is part of a gene in gene prediction problems (Culotta et al., 2005), table extraction (Pinto et al., 2003), and object recognition (Quattoni et al., 2004). A main reason for the popularity and empirical successes of CRFs is the flexibility to incorporate arbitrary dependencies on the observations. Such dependencies may be non-independent and overlapping. Such flexibility is not present in generative models like Markov models, in which strong independence assumptions are required.

We first discuss the connections between CRFs and several other models, showing that CRFs can be used as a very general class of models that possesses the modeling power of MaxEnt classifiers, MEMMs, naive Bayes and Markov chains.

We then describe CRFs as models on variables which can be specified by two equivalent views: a model satisfying independence relationships that can be conveniently

described as Markov assumptions associated with a graph, or a model possessing a density function which are products of factors defined on subsets of the variables. This perspective allows conditional independence to be easily discovered with the help of graphs. This is useful because conditional independence properties are useful for using sampling techniques such as Gibbs sampling to compute marginals. In addition, this perspective makes it convenient to study general efficient marginal computation algorithms. For example, the message passing algorithm (or belief propagation, or sum-product algorithm) (Huang and Darwiche, 1996) is a general algorithm which can be conveniently described using the graphical structures.

3.5.1 Connections with Other Models

We have seen some special cases of CRFs. MEMMs are CRFs with the parameters for different components estimated separately. MaxEnt classifiers are obtained by restricting sequence length to 1. Logistic regression can be obtained as a special case of the MaxEnt classifier by choosing $f_{i,K}(x, y) = 0$ for all i in the MaxEnt classifiers. Another special case of the MaxEnt classifier is the majority vote algorithm which predicts the most common class for any x in the training data, but randomly predict a class if x is not seen before. To see this, for each x in the training data, construct a feature $f_{x',y'}(x, y) = \mathrm{I}(x = x' \text{ and } y = y')$ for each class y. Setting the parameter for $f_{x',y'}$ as $\lambda_{x',y'} = \ln \tilde{p}(y'|x')$, then we obtain the majority vote algorithm.

The conditional distribution of Markov models is a family of CRFs with the parameters being constrained to be non-positive. If i.i.d. observations are generated by a Markov model, and CRFs are learned using MLE, then the CRFs converge to the true conditional distribution. Since naive Bayes and MaxEnt classifiers (or logistic regression) are obtained from Markov models and CRFs respectively by restricting the sequence length to be 1, the same remark holds for them.

Note that logistic regression or MaxEnt classifiers have linear decision boundaries

in the attribute space, thus they are generally different from decision trees constructed using the same set of attributes. However, using the paths in the decision tree as features, then there is a MaxEnt model which is equivalent to the decision tree.

3.5.2 Undirected Graphical Models

CRFs can be characterized as distributions satisfying certain conditional independence assumptions.

The key idea is the following way to visualize a probability distribution over a collection of random variables: Decompose the distribution into factors which contain as few random variables as possible, and construct a graph with the random variables as nodes, and link every pair of random variables that are in the same factor. This graphical representation turns out to be convenient for deriving conditional independences about the given distribution. One simple conditional independence that is easy to verify is that a variable is conditionally independent of other variables given its neighbors. Hammersley and Clifford showed the surprising result that distributions satisfying such conditional independences can be expressed as products of factors over the cliques (Hammersley and Clifford, 1971).

A.*Factorization over Cliques and Markov Independence.* Let $X = \{X_1, \ldots, X_n\}$ be a set of random variable. Let $V = \{1, \ldots, n\}$, and $G = (V, E)$ be an undirected graph with V as the vertex set and E as the edge set. For any $I \subseteq V$, V can be partitioned into three disjoint set, I itself, I's *boundary* or *Markov blanket* ∂I consisting of vertices which are not in I but connected to at least one vertex in I, and $\nsim I = V - I - \partial I$. Let X_I denote the subset of X with index set I. Let Z denote a subset of X. The boundary ∂Z of Z is $X_{\partial I}$, where I is the index set of Z. A probability distribution P over X is said to be *Markovian* for Z on G if $P(Z|X - Z) = P(Z|\partial Z)$. If P is Markovian for any $Z \subseteq X$, then P is said to be *Markovian* on G, and (P, G) is called a *Markov random field* (MRF).

Throughout this section, we shall only deal with positive distributions. Hammersley and Clifford proved the following fundamental result for Markov random fields: (P, G) is an MRF iff P is the product of factors defined over the cliques of G.

Theorem 27. *(Hammersley-Clifford (Hammersley and Clifford, 1971)) Let $\mathcal{C}$ be the set of cliques of G. Then (P, G) is an MRF iff there exist positive functions $\{g_C(X_C) : C \in \mathcal{C}\}$ such that $P(X) = \prod_{C \in \mathcal{C}} g_C(X_C)$.*

We present a simplified proof based on (Hammersley and Clifford, 1971). The Markovian property can be stated as an invariant property of the log-likelihood function $L(X) = \log P(X)$, as follows. For each X_i, we arbitrarily choose one of its values as its reference value. For any instantiation z of some $Z \subseteq X$, z^I denotes the instantiation obtained from z by setting values of random variables with indices in I to their reference values. Define a normalization operator N_I by $N_I(R(x)) = R(x^I)$ for all x, where R is an arbitrary function on X. We overload the notation x_I: If x is mentioned as an instantiation for X, then x_I denotes its instantiation for random variables with indices in I, otherwise x_I denote an instantiation for X_I. The meaning will be clear from the context. In addition, if I_1 and I_2 are disjoint, then $x_{I_1} \cup x_{I_2}$ denotes the combined instantiation for $X_{I_1 \cup I_2}$. The invariant property we need is as follows.

Proposition 28. *$P(X)$ is Markovian for X_I on G iff $L(X)$ is an invariant of $N_I + N_{\nsim I} - N_{I \cup \nsim I}$ for any $I \subseteq V$.*

Proof. Note that $P(X)$ is Markovian for X_I on G iff for any instantiation x of X, we have

$$\frac{P(x_I, x_{\partial I}, x_{\nsim I})}{P(x_{\partial I}, x_{\nsim I})} = \frac{P(x_I, x_{\partial I}, x^{\nsim I}_{\nsim I})}{P(x_{\partial I}, x^{\nsim I}_{\nsim I})}, \frac{P(x^I_I, x_{\partial I}, x_{\nsim I})}{P(x_{\partial I}, x_{\nsim I})} = \frac{P(x^I_I, x_{\partial I}, x^{\nsim I}_{\nsim I})}{P(x_{\partial I}, x^{\nsim I}_{\nsim I})}$$

These two equations are equivalent to

$$\begin{aligned}
& P(x_I, x_{\partial I}, x_{\nsim I})/P(x^I_I, x_{\partial I}, x_{\nsim I}) &=& P(x_I, x_{\partial I}, x^{\nsim I}_{\nsim I})/P(x^I_I, x_{\partial I}, x^{\nsim I}_{\nsim I}) \\
\Leftrightarrow\ & L(x_I, x_{\partial I}, x_{\nsim I}) - L(x^I_I, x_{\partial I}, x_{\nsim I}) &=& L(x_I, x_{\partial I}, x^{\nsim I}_{\nsim I}) - L(x^I_I, x_{\partial I}, x^{\nsim I}_{\nsim I}) \\
\Leftrightarrow\ & L(x) - N_I L(x) &=& N_{\nsim I} L(x) - N_{I \cup \nsim I} L(x) \\
\Leftrightarrow\ & L(x) &=& (N_I + N_{\nsim I} - N_{I \cup \nsim I}) L(x).
\end{aligned}$$

■

We now give the proof for Theorem 27 below.

Proof. ($\Longleftarrow$) Suppose $P(X)$ is of the form $\prod_C g_C(X_C)$, then for any $I \subseteq V$, for any x, $P(x)$ is the product of $\prod_{C \cap I = \emptyset} g_C(x_C)$ and $\prod_{C \cap I \neq \emptyset} g_C(x_C)$. The first factor does not depend on x_I, while the second factor depends on $x_{I \cap \partial I}$ because the completeness of C and $C \cap I \neq \emptyset$ implies that every vertex of C is either in I or ∂I. Thus the second factor can be written as $\prod_{C \subseteq I \cup \partial I} g_C(x_C)$. Hence for any $x_{\partial I}$, $x_{\nsim I}$, for any two instantiations x_I, x'_I of X_I, we have

$$P(x_I, x_{\partial I}, x_{\nsim I})/P(x'_I, x_{\partial I}, x_{\nsim I}) \;=\; \textstyle\prod_{C \subseteq I \cup \partial I} g_C((x_I \cup x_{\partial I})_C) / \prod_{C \subseteq I \cup \partial I} g_C((x'_I \cup x_{\partial I})_C)$$

Using the above equation, we have

$$\begin{aligned}
P(x_I | x_{\partial I}, x_{\nsim I}) &= P(x_I, x_{\partial I}, x_{\nsim I}) / \textstyle\sum_{x'_I} P(x'_I, x_{\partial I}, x_{\nsim I}) \\
&= \textstyle\prod_{C \subseteq I \cup \partial I} g_C((x_I \cup x_{\partial I})_C) / \sum_{x'_I} \prod_{C \subseteq I \cup \partial I} g_C((x'_I \cup x_{\partial I})_C).
\end{aligned}$$

Note that the above equation shows that the value of $P(x_I | x_{V-I})$ does not depend on $x_{\nsim I}$. Hence $P(X_I | X_{V-I}) = P(X_I | X_{\partial I})$.

($\Longrightarrow$) Suppose (P, G) is an MRF then

$$\begin{aligned} L &= \textstyle\prod_{i\in V}[N_{\{i\}} + N_{\sim\{i\}} - N_{\{i\}\cup\sim\{i\}}]L \\ &= \textstyle\prod_{i\in V}[N_{\sim\{i\}}(1 - N_{\{i\}}) + N_{\{i\}}]L \\ &= \textstyle\sum_{I\subseteq V}\prod_{i\in I}[N_{\sim\{i\}}(1 - N_{\{i\}})]\prod_{i\in V-I} N_{\{i\}}L. \end{aligned} \tag{3.5.1}$$

If I is not a clique of G, then there exists $i, j \in I$ which are not neighbors. In this case, we have $N_{\sim\{i\}}(1 - N_{\{j\}})L = (N_{\sim\{i\}} - N_{\{j\}}N_{\sim\{i\}})L = (N_{\sim\{i\}} - N_{\sim\{i\}})L = 0$. However $N_{\sim\{i\}}(1 - N_{\{j\}})$ appears in the product term for I, thus the product term for I vanishes.

Hence, $L = \sum_{C\subseteq V}\prod_{i\in I}[N_{\sim\{i\}}(1 - N_{\{i\}})]\prod_{i\in V-C} N_{\{i\}}L$. In addition, observe that $\prod_{i\in V-C} N_{\{i\}})L(x) = L(x^{V-C})$ depends on x_C only, thus $\prod_{i\in I}[N_{\sim\{i\}}(1-N_{\{i\}})]\prod_{i\in V-C} N_{\{i\}}L$ is equal to some function $h_C(X_C)$. It follows that $L(X) = \sum_{C\subseteq V} h_C(X_C)$ and $P(X) = \prod_{C\subseteq V} e^{h_C(X_C)}$. The proof is completed by letting $g_C(X_C) = h_C(X_C)$, ■

B.*Additiveness of local Markovian property.* Hammersley and Clifford (1971) showed that if P is Markovian for all X_i on G, then P is Markovian on G. We prove the following additiveness property for being locally Markovian. The above result follows as a simple consequence.

Proposition 29. *If P is Markovian at $I_1, \ldots, I_k$ on G, then P is Markovian at $I_1 \cup \ldots \cup I_k$ on G.*

Proof. Let N_{I_1,I_2} denote $N_{I_1} + N_{I_2} - N_{I_1 \cup I_2}$. If $I_1, I_2 \subseteq I$, then $\nsim I \subseteq \nsim I_2$, and

$$
\begin{aligned}
& N_{I_1,\nsim I} N_{I_2,\nsim I_2} \\
= \; & (N_{I_1} + N_{\nsim I} - N_{I_1 \cup \nsim I})(N_{I_2} + N_{\nsim I_2} - N_{I_2 \cup \nsim I_2}) \\
= \; & (N_{I_1 \cup I_2} + \cancel{N_{I_1 \cup \nsim I_2}} - \cancel{N_{I_1 \cup I_2 \cup \nsim I_2}}) + (N_{\nsim I \cup I_2} + N_{\nsim I_2} - N_{I_2 \cup \nsim I_2}) \\
& -(N_{I_1 \cup I_2 \cup \nsim I} + \cancel{N_{I_1 \cup \nsim I_2}} - \cancel{N_{I_1 \cup I_2 \cup \nsim I_2}}) \\
= \; & N_{I_1 \cup I_2} + N_{\nsim I \cup I_2} + N_{\nsim I_2} - N_{I_2 \cup \nsim I_2} - N_{I_1 \cup I_2 \cup \nsim I} \\
= \; & (N_{I_1 \cup I_2} + \cancel{N_{\nsim I}} - N_{I_1 \cup I_2 \cup \nsim I}) + (\cancel{N_{I_2}} + N_{\nsim I_2} - N_{I_2 \cup \nsim I_2}) - (\cancel{N_{I_2}} + \cancel{N_{\nsim I}} - N_{I_2 \cup \nsim I}) \\
= \; & N_{I_1 \cup I_2, \nsim I} + N_{I_2, \nsim I_2} - N_{I_2, \nsim I}.
\end{aligned}
$$

For any $I_1, I_2 \subseteq I$, we have

$$
\begin{aligned}
(N_{I,\nsim I} - N_{I_1,\nsim I}) N_{I_2,\nsim I_2} &= N_{I \cup I_2,\nsim I} + N_{I_2,\nsim I_2} - N_{I_2,\nsim I} - (N_{I_1 \cup I_2,\nsim I} + N_{I_2,\nsim I_2} - N_{I_2,\nsim I}) \\
&= N_{I,\nsim I} - N_{I_1 \cup I_2,\nsim I}.
\end{aligned}
$$

Now let $I = I_1 \cup \ldots \cup I_k$. If $P(X)$ is Markovian for all I_j on G, then $L(X)$ is invariant of $N_{I_j}, \nsim I_j$ for all j, and thus we have

$$
\begin{aligned}
N_{I,\nsim I} L - L &= (N_{I,\nsim I} - N_{\emptyset,I}) L \\
&= (N_{I,\nsim I} - N_{\emptyset,I}) \prod_{i \in I} N_{\{i\},\nsim\{i\}} L \\
&= (N_{I,\nsim I} - N_{I,\nsim I}) L \\
&= 0.
\end{aligned}
$$

■

C. *Conditional random fields.* Hammersley-Clifford's theorem clearly holds for conditional distributions of a collection of variables as well. A CRF is then defined as a

pair (P, G) where $P(Y|X)$ satisfies the conditional Markov property with respect to G. Note that this definition includes the CRFs in Example 15 as special cases.

D.*Connections to directed graphical models.* For directed graphical models, the Markov blanket of a node consists of its parent, its children and any other parents of its children. The Markov blanket of a set of nodes I is the union of the Markov blankets of its elements but with nodes in I removed. The proof of Proposition 28 only requires the Markov blanket of a set to be disjoint with it, thus it holds for directed graphical models as well. Similarly, the proof of Proposition 29 only requires the definition of Markov blanket to satisfy the additional property that $\nsim I \subseteq \nsim I_1$ whenever $I_1 \subseteq I$, but this property holds for the directed case as well, thus it also holds that the property of being locally Markovian is additive. There is also a factorization result similar to that of Hammersley and Clifford: Let G be a DAG, and P be a positive distribution. Then P is Markovian on G iff there exists positive probability functions $f(x_i|Pa(X_i))$ such that $P(X) = \prod_{i=1}^{n} f(X_i|Pa(X_i))$. However, the proof of Theorem 27 does not have an immediate analogue in this case.

3.5.3 Inference

We are often interested in computing the most likely configuration for all variables or single variables for MRFs or CRFs. There is a well-known connection between the most likely configuration for all variables and the most likely configurations for single variables.

Proposition 30. *Suppose* $P(X_1, \ldots, X_n|t, \theta) \propto e^{\theta \cdot T(X_1,\ldots,X_n)/t}$, *where* t *is often called a temperature parameter. Assume there is a unique most likely global configuration at temperature* $t = 1$, *and denote it by*

$$(x_1^*, ..., x_n^*) = \arg \max_{x_1,\ldots,x_n} P(x_1, \ldots, x_n|t = 1, \theta), \tag{3.5.2}$$

Denote the most likely (breaking ties arbitrarily) configuration for a single variable at temperature t as

$$x'_{i,t} = \arg\max_{x_i} P(x_i|t, \theta). \tag{3.5.3}$$

Then when $t > 0$ is sufficiently small, $x^*_i = x'_{i,t}$ for all i.

The above result implies that if we have an algorithm to compute the marginals for single variables, then by making the temperature small enough, we can recover the most likely global configuration.

For the case when the graphical structure is a linear chain, computing the most likely configurations for all the variables and single variables can be done efficiently. In this case, the distribution can be written as (omitting the observed variables for the case of CRFs)

$$P(X_1, \ldots, X_n) = C\varphi_{1,2}(X_1, X_2)\varphi_{2,3}(X_2, X_3)\ldots\varphi_{n-1,n}(X_{n-1}, X_n), \tag{3.5.4}$$

where C is some constant. Thus we have

$$\begin{aligned}
P(x_n) &= C\textstyle\sum_{x_1,\ldots,x_{n-1}} P(x_1, \ldots, x_n) \\
&= C\textstyle\sum_{x_2,\ldots,x_{n-1}} [\sum_{x_1} \varphi_{1,2}(x_1, x_2)]\varphi_{2,3}(x_2, x_3)\ldots\varphi_{n-1,n}(x_{n-1}, x_n) \\
&= C\textstyle\sum_{x_2,\ldots,x_{n_1}} \alpha_2(x_2)\varphi_{2,3}(x_2, x_3)]\varphi_{3,4}(x_3, x_4)\ldots\varphi_{n-1,n}(x_{n-1}, x_n) \\
&= C\textstyle\sum_{x_3,\ldots,x_{n_1}} [\sum_{x_2} \alpha_2(x_2)\varphi_{2,3}(x_2, x_3)]\varphi_{3,4}(x_3, x_4)\ldots\varphi_{n-1,n}(x_{n-1}, x_n) \\
&= C\textstyle\sum_{x_3,\ldots,x_{n_1}} \alpha_3(x_2)\varphi_{3,4}(x_3, x_4)\ldots\varphi_{n-1,n}(x_{n-1}, x_n) \\
&\ldots \\
&= C\alpha_n(x_n).
\end{aligned}$$

The marginal $P(X_n)$ can now be expressed as

$$P(x_n) = \alpha_n(x_n)/\textstyle\sum_{x'_n} \alpha_n(x'_n). \tag{3.5.5}$$

In the above equations, $\alpha_1(x_1) = 1$, and for $t > 1$,

$$\alpha_t(x_t) = \sum_{x_{t-1}} \alpha_{t-1}(x_{t-1})\varphi(x_{t-1}, x_t). \tag{3.5.6}$$

α_i's are called the *forward variables* and a more intuitive definition of it is

$$\alpha_t(x_t) = \sum_{x_1,\ldots,x_{t-1}} \varphi_{1,2}(x_1, x_2)\ldots\varphi_{t-1,t}(x_{t-1}, x_t). \tag{3.5.7}$$

The above algorithm is known as the *variable elimination* algorithm (Zhang and Poole, 1994). The time complexity is $O(nm^2)$ if each variable can take at most m values.

Variable elimination can be used to compute all $P(X_i)$'s separately in a total of $O(n^2m^2)$ time. This can be sped up using the *forward-backward algorithm* (Rabiner, 1989). Define the backward variable

$$\beta_t(x_t) = \sum_{x_{t+1},\ldots,x_n} \varphi_{t,t+1}(x_t, x_{t+1})\ldots\varphi_{n-1,n}(x_{n-1}, x_n), \tag{3.5.8}$$

Then we have a similar DP algorithm to compute the backward variables by noting that $\beta_n(x_n) = 1$ and

$$\beta_t(x_t) = \sum_{x_{t+1}} \varphi(x_t, x_{t+1})\beta_{t+1}(x_{t+1}). \tag{3.5.9}$$

It is easy to verify that

$$P(x_t, x_{t+1}) = C\alpha_t(x_t)\varphi_{t,t+1}(x_t, x_{t+1})\beta_{t+1}(x_{t+1}). \tag{3.5.10}$$

Marginalizing away one of the variables, then we get the required marginals. The time complexity is $O(nm^2)$.

The most likely configurations for all variables can be found by replacing the sum operators using max operators in variable elimination.

Inference for general graphical structure is known to be NP-hard (Istrail, 2000),

but inference over tree structures can be done using belief propagation (Pearl, 1982). For general graphical structure, the junction tree algorithm (a.k.a. the clique tree algorithm) can be used to convert the inference problem to one for a tree structure. See (Huang and Darwiche, 1996) for a procedural guide for the case of Bayesian networks, and (Kschischang et al., 2001) for a presentation of the algorithm using factor-graphs. Alternatively, loopy belief propagation can often be used to give good probability estimates as well (Yedidia et al., 2003).

Chapter 4

Sparse High-order CRFs for Sequence Labeling

As we have pointed out in Section 3.5, CRFs have been successfully applied in various problems, including named-entity recognition, handwriting recognition, gene prediction, table extraction, and object recognition. However, inference for general CRFs are intractable. While a significant part of the modeling power of CRF depends on its flexibility to incorporate arbitrary structural dependencies, the complexity of general inference algorithms blows up quickly as the complexity of structural dependencies being captured increase. For example, consider order k dependencies, which are dependencies between $k+1$ consecutive labels. While capturing high-order dependencies between consecutive characters in handwritting recognition (Kassel., 1995) is useful, a general inference algorithms like the junction tree algorithm has time complexity exponential in the maximum order (Huang and Darwiche, 1996). Another example where structural dependency is useful is in labeling multiple activities (Patterson et al., 2005). There are often patterns in which activities can happen, for example, one activity may exclude another, or imply another. But capturing such dependencies leads to computationally hard problems. For example, FCRFs can be used to capture such

dependencies, but the time complexity scales exponentially as the number of activities increases. Thus although structural dependencies among labels is an important source of information, and CRFs can be used to capture arbitrary dependencies of the labels on the observations, typically only first-order dependencies, that is, dependencies between adjacent labels are considered.

Since restricting the kind of dependencies a model uses can lead to significant loss of information, approximation algorithms have often been used to handle complex dependencies. However, while approximation algorithms have been successfully applied, they have also been seen to give results which are not really related to the true ones Murphy et al. (1999). This motivates us to search for efficient exact algorithms for sub-classes of CRFs. In particular, in this chapter and the next, we shall demonstrate that such efficient exact algorithms exist for two classes of CRFs with sparse potential funtions.

This chapter presents efficient inference and learning algorithms for high-order CRFs, that is, CRFs with high-order dependencies. The algorithms exploit the observation that generally the set of observed label patterns is sparse, that is, the number of observed label patterns is much smaller than the number of possible patterns. The time complexity scales polynomially in the number of observed label patterns. The results presented here describes and extends our previous work on high-order CRFs (Ye et al., 2009).

Here is an outline of this chapter. Section 4.1 first reviews some related works on capturing long-range dependencies. Section 4.2 presents the form of high-order features used. Section 4.3 discusses the effect of omitting inactive features in a CRF, and provides a justification for using only seen label patterns in high-order features. Section 4.4 derives efficient algorithms for computing marginals and Viterbi parses for sparse high-order CRFs. Section 4.5 presents how maximum likelihood learning can be done by using gradient descent algorithms. Section 4.6 describes some extensions of

the algorithms presented. Section 4.7 demonstrates that high-order features can lead to substantial performance improvements for some problems and discuss conditions under which high-order features can be effective.

4.1 Long-range Dependencies

A CRF can be used to capture arbitrary dependencies among components of $\mathbf{x}$ and $\mathbf{y}$, in practice, this flexibility of the CRF is not fully exploited as inference in Markov models is NP-hard in general (Istrail, 2000), and can only be performed efficiently for special cases such as linear chains. As such, most applications involving CRFs are limited to some *tractable* Markov models. This observation also applies to other structured prediction methods such as structured support vector machines (Taskar et al., 2003; Tsochantaridis et al., 2004).

A commonly used inference algorithm for CRF is the clique tree algorithm (Huang and Darwiche, 1996). However, with a feature depending on k consecutive labels, the running time will be exponential in k. When only a small number of patterns are used, our algorithm achieves efficiency by maintaining only information related to a few occurred patterns, while previous algorithms maintain information about all (exponentially many) possible patterns.

Long distance dependencies can also be captured using hierarchical models such as Hierarchical Hidden Markov Model (HHMM) (Shai Fine and Tishby, 1998) or Probabilistic Context Free Grammar (PCFG) (Jelinek et al., 1992). The time complexity of inference in an HHMM is $O(\min\{nl^3, n^2l\})$ (Shai Fine and Tishby, 1998; Murphy and Paskin, 2002), where n is the number of states and l is the length of the sequence. Discriminative versions such as hierarchical CRF has also been studied (Truyen et al., 2008). Inference in PCFG and its discriminative version can also be efficiently done in $O(ml^3)$ where m is the number of productions in the grammar (Jelinek et al., 1992).

These methods are able to capture dependencies of arbitrary lengths, unlike k-order Markov chains. However, to do efficient learning with these methods, the hierarchical structure of the examples need to be provided. For example, if we use PCFG to do named entity recognition, we need to provide the parse trees for efficient learning; providing the named entity labels for each word is not sufficient. Hence, a training set that has not been labeled with hierarchical labels will need to be relabeled before it can be trained efficiently. Alternatively, methods that employ hidden variables can be used (e.g. to infer the hidden parse tree) but the optimization problem is no longer convex and local optima can sometimes be a problem. Using high-order features captures less expressive form of dependencies than these models but allows efficient learning without relabeling the training set with hierarchical labels.

Qian et al. (2009) independently presented algorithms for a larger class of sparse high-order CRFs, which can require exponential time, and they did not identify a class of high-order CRFs with polynomial time algorithms. Their algorithm relies on grouping the states and the transitions with the same potential functions, and performing inference on the grouped states. Our algorithm also uses grouping to yield a compact representation for efficient inference, but exploits the structure of the particular form of high-order features for such grouping.

4.2 High-order Features

We shall use $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ (with or without decorations) to denote respectively an observation sequence of length T, a label sequence of length T, and an arbitrary label sequence. The function $|\cdot|$ denotes the length of any sequence. The set of labels is $\mathcal{Y} = \{1, \ldots, n\}$. If $\mathbf{z} = (y_1, \ldots, y_t)$, then $\mathbf{z}_{i:j}$ denotes $(y_i, \ldots, y_j)$. When $j < i$, $\mathbf{z}_{i:j}$ is the empty sequence (denoted by ϵ). Let the features being considered be $f_1, \ldots, f_m$. Each feature f_i is

associated with a label sequence $\mathbf{z}^i$, called f_i's *label pattern*, and f_i has the form

$$f_i(\mathbf{x}, \mathbf{y}, t) \quad = \quad \mathrm{I}(\mathbf{y}_{t-|\mathbf{z}^i|+1:t} = \mathbf{z}^i) g_i(\mathbf{x}, t). \tag{4.2.1}$$

We call f_i a feature of *order* $|\mathbf{z}^i| - 1$. To illustrate, consider the problem of named entity recognition. The observations $\mathbf{x} = (x_1, \ldots, x_T)$ may be a word sequence; $g_i(\mathbf{x}, t)$ may be an indicator function for whether x_t is capitalized or may output a precomputed term weight if x_t matches a particular word; and $\mathbf{z}^i$ may be a sequence of two labels, such as (*person*, *organization*) for the named entity recognition task, giving a feature of order one.

Recall that a CRF defines conditional probability distributions $P(\mathbf{y}|\mathbf{x}) = Z_{\mathbf{x}}(\mathbf{y})/Z_{\mathbf{x}}$, where $Z_{\mathbf{x}}(\mathbf{y}) = \exp(\sum_{i=1}^{m} \sum_{t=|\mathbf{z}^i|}^{T} \lambda_i f_i(\mathbf{x}, \mathbf{y}, t))$, and $Z_{\mathbf{x}} = \sum_{\mathbf{y}} Z_{\mathbf{x}}(\mathbf{y})$. The normalization factor $Z_{\mathbf{x}}$ is called the *partition function*. Most of the time we shall be concerned with only a fixed sequence $\mathbf{x}$, thus we often simply write $Z(\mathbf{y})$ and Z for $Z_{\mathbf{x}}(\mathbf{y})$ and $Z_{\mathbf{x}}$. The same convention applies to notations defined later.

4.3 Sparsity

The effectiveness of our algorithms for high order CRFs relies on two assumptions. One is that the set of observed label patterns is sparse, that is, the number of observed label patterns is much smaller than the number of possible label patterns. The other is that omitting unseen label patterns does not have adverse effect on performance. The first assumption generally holds for real datasets. We discuss the second assumption.

An attribute or a feature is said to be *inactive* if its value is always 0 in the training data, and active otherwise. Recall that as defined in Section 2.3.1 an attribute is a function of the observation, while a feature is a function of both the observation and label. Hence omitting inactive attributes and omitting inactive features are not the same, because one attribute may correspond to different features. For example, in

logistic regression, the i-th attribute is associated with features of the form $f_{i,y'}(x, y)$.

If an attribute is inactive, then the models learned with and without the attribute are often the same. For example, this is true for SVMs, and the generalized representer theorem says that this is true for a large class of algorithms (Schölkopf et al., 2001). For the case of CRFs, it is also easy to show that if training is done by maximizing likelihood regularized with a Gaussian prior, then features using an inactive attribute will have weight 0. Hence omitting inactive attributes lead to a more compact model without hurting performance.

However, omitting an inactive feature generally do not lead to the same models. For CRFs, the weights of inactive features in an MLE are negative-infinity. Hence, inactive features cannot be activated in the most probable explanations, that is, Viterbi parses, determined by such MLEs. On the other hand, omitting a feature is equivalent to forcing its weight to be 0. If the MLE for a CRF with inactive features omitted is used to determine the Viterbi parse of a sequence, then inactive features may be active. So using maximum likelihood estimation, the sparse model tolerates the activation of inactive features, while the nonsparse model does not.

In the case our high-order features, using unobserved label patterns in the features forbids these label patterns from occurring in the Viterbi parses. This can easily cause overfitting, especially when there are only a few examples. Omitting inactive features avoids this problem.

Certainly, the above discussion does not guarantee that omitting inactive features leads to a model with better performance. In addition, with regularization, the possibility of overfitting caused by using unobserved label patterns is reduced.

In practice, we observed that omitting inactive features may or may not lead to better performance, but generally the difference is not large, so whether we should use a sparse model or a dense one may not be the most important – the more important thing is to first choose good features based on domain knowledge. Then one can use

a sparse model if computation resource is limited, since omitting inactive features can effectively reduce the time and space complexity.

4.4 Viterbi Parses and Marginals

In this section, we describe the algorithms for computing the partition function, the marginals and the most likely label sequence for high-order CRFs. We give rough polynomial time complexity bounds to give an idea of the effectiveness of the algorithms. These bounds are pessimistic compared to practical performance of the algorithms. It can also be verified that the algorithms for linear chain CRF (Lafferty et al., 2001) are special cases of our algorithms when only zero-th and first order features are considered.

We shall work with three sets: the *pattern set* $\mathcal{Z}$, the *forward-state set* $\mathcal{P}$ and the *backward-state set* $\mathcal{S}$. The pattern set, $\mathcal{Z}$, is the set of distinct label patterns used in the m features. The forward-state set, $\mathcal{P}$, consists of all labels and all proper prefixes (including ϵ) of label patterns. Similarly, $\mathcal{S}$ consists of the labels and all proper suffixes (including ϵ) of label patterns. Unless otherwise stated, $\mathbf{p}$ / $\mathbf{s}$ (decorated or not) denotes an element of $\mathcal{P}$ / $\mathcal{S}$.

We can use the forward / backward states to induce partitions of label sequences as follows. For $\mathbf{p} \in \mathcal{P}$, a label sequence $\mathbf{z}$ is said to be in the forward state $\mathbf{p}$, denoted by $\mathbf{z} \overset{p}{\in} \mathbf{p}$, if $\mathbf{p}$ is the longest element in $\mathcal{P}$ such that $\mathbf{z}$ *ends* with $\mathbf{p}$. We adopt the notation $\overset{p}{\in}$ to highlight the fact that each forward state $\mathbf{p}$ can be considered as the set of all label sequences which are in $\mathbf{p}$. Similarly, for $\mathbf{s} \in \mathcal{S}$, a label sequence $\mathbf{z}$ is said to be in the backward state $\mathbf{s}$, denoted by $\mathbf{z} \overset{s}{\in} \mathbf{s}$, if $\mathbf{s}$ is the longest element in $\mathcal{S}$ such that $\mathbf{z}$ *starts* with $\mathbf{s}$. The following proposition shows that the set of forward/backward states indeed induce a partition of the label sequences.

Proposition 31. *Any* $\mathbf{z}$ *is in one and only one forward / backward state.*

Proof. Consider the forward state set $\mathcal{P}$. For any $\mathbf{z}$, consider the set of all $\mathbf{p} \in \mathcal{P}$

such that $\mathbf{z}$ ends with $\mathbf{p}$. This set is nonempty because $\mathbf{z}$ ends with ϵ. In addition all elements in this set are of different lengths, thus there is a unique longest element. That is, $\mathbf{z}$ is in one and only one forward state.

The backward case is similar. ■

We define state transition functions for the forward and backward states:

$$T^p(\mathbf{p}, y) \stackrel{\text{def}}{=} \mathbf{p}' \quad \text{if } \mathbf{p}y \stackrel{p}{\in} \mathbf{p}', \tag{4.4.1}$$

$$T^s(\mathbf{s}, y) \stackrel{\text{def}}{=} \mathbf{s}' \quad \text{if } y\mathbf{s} \stackrel{s}{\in} \mathbf{s}', \tag{4.4.2}$$

where $\mathbf{p}^i y$ and $y\mathbf{s}$ denote concatenations of two strings. By Proposition 31, we know that $\mathbf{p}'$ and $\mathbf{s}'$ exist and are unique. Equivalently,

Proposition 32. *T^p and T^s are well defined on $\mathcal{P} \times \mathcal{Y}$ and $\mathcal{S} \times \mathcal{Y}$ respectively.*

We will generally drop the p and s in the notations $\stackrel{p}{\in}$, $\stackrel{s}{\in}$, T^p, T^s, as it is usually clear whether we are working with forward states or backward states.

Finally, the subsequence relationship defined below are used when combining forward and backward variables to compute marginals. Let $\mathbf{z} \subseteq \mathbf{z}'$ denote that $\mathbf{z}$ is a subsequence of $\mathbf{z}'$, $\mathbf{z} \subset \mathbf{z}'$ denote that $\mathbf{z}$ is a subsequence of $\mathbf{z}'_{2:|\mathbf{z}'|-1}$. The addition of subscript j in $\subseteq_j$ and $\subset_j$ indicates that $\mathbf{z}$ ends at position j in $\mathbf{z}'$ as well.

In the following, we consider how to compute several key quantities for a fixed observation sequence $\mathbf{x}$.

4.4.1 The Forward and Backward Variables

We now define forward vector α and backward vector β and give dynamic programming algorithms for computing them. We shall first describe the algorithm for the forward vector. The algorithm for the backward vector is similar.

A forward variable is the sum of scores of all partial sequences of the same length and in the same forward state. Formally, consider a label sequence $\mathbf{y}$ for $\mathbf{x}$, define the prefix score function

$$P(\mathbf{y}, t) \stackrel{\text{def}}{=} \exp(\sum_{i=1}^{m} \sum_{t'=|\mathbf{z}^i|}^{t} \lambda_i f_i(\mathbf{x}, \mathbf{y}, t')). \tag{4.4.3}$$

Note that if $\mathbf{z} = \mathbf{y}_{1:t} = \mathbf{y}'_{1:t}$, then $P(\mathbf{y}, t) = P(\mathbf{y}', t)$, thus we can define $P(\mathbf{z}) = P(\mathbf{y}, t)$. The *forward variable* $\alpha(t, \mathbf{p})$ is defined as $\alpha(t, \mathbf{p}) \stackrel{\text{def}}{=} \sum_{|\mathbf{z}|=t, \mathbf{z} \in \mathbf{p}} P(\mathbf{z})$. Each vector $\alpha(t, \cdot)$ is of dimension $|\mathcal{P}|$.

The key ingredients to the DP algorithm for computing α are the following two properties.

Proposition 33. *(a) (Factorization) If $\mathbf{z} \in \mathbf{p}$, then*

$$P(\mathbf{z}y) = \Psi^p(|\mathbf{z}y|, \mathbf{p}y) P(\mathbf{z}), \quad \textit{where} \tag{4.4.4}$$

$$\Psi^p(t, \mathbf{z}) \stackrel{\textit{def}}{=} \exp(\textstyle\sum_{\mathbf{z}^i \textit{ is a suffix of } \mathbf{z}} \lambda_i g_i(\mathbf{x}, t)) \tag{4.4.5}$$

(b) (Transition) For any $\mathbf{z}$ and y, if $\mathbf{z} \in \mathbf{p}$, then $\mathbf{z}y \in T(\mathbf{p}, y)$.

Proof. (a) Let $\mathbf{y}$ be an arbitrary length $|\mathbf{x}|$ label sequence extending $\mathbf{z}y$, $t = |\mathbf{z}y|$, then by the definition of the prefix score function,

$$P(\mathbf{z}y)/P(\mathbf{z}) = \exp(\textstyle\sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}, \mathbf{y}, t)).$$

By the definition of f_i in Eq. 4.2.1, $f_i(\mathbf{x}, \mathbf{y}, t)$ is 0 if $\mathbf{z}^i$ is not a suffix of $\mathbf{z}y$, and it is equal to $g_i(\mathbf{x}, t)$ otherwise. Hence

$$P(\mathbf{z}y)/P(\mathbf{z}) = \exp(\textstyle\sum_{\mathbf{z}^i \text{ is a suffix of } \mathbf{z}y} \lambda_i g_i(\mathbf{x}, t)).$$

Suppose $\mathbf{z}^i$ is a suffix of $\mathbf{z}y$ but not a suffix of $\mathbf{p}y$. Let $\mathbf{p}'$ be the sequence obtained by removing last label of $\mathbf{z}^i$, then $\mathbf{p}'$ is a forward state, but $\mathbf{z}$ ends with $\mathbf{p}'$ and $\mathbf{p}'$ is longer than $\mathbf{p}$. Hence $\mathbf{z}$ is not in the forward state $\mathbf{p}$, a contradiction. Thus $\mathbf{z}^i$ must be a suffix of $\mathbf{p}y$, and

$$P(\mathbf{z}y)/P(\mathbf{z}) = \exp(\textstyle\sum_{\mathbf{z}^i \text{ is a suffix of } \mathbf{p}y} \lambda_i g_i(\mathbf{x}, t)).$$

(b) Consider the set of forward states which $\mathbf{z}y$ ends with. None of such forward state can be longer than $\mathbf{p}y$ because otherwise $\mathbf{p}$ is no longer the longest forward state $\mathbf{z}$ ends with. Hence all these forward states must be suffixes of $\mathbf{p}y$, and the longest one must be $T(\mathbf{p}, y)$. ■

From the above two properties, we can derive the following DP algorithm for α.

Theorem 34. *The forward variable satisfies* $\alpha(0, \epsilon) = 1$, $\alpha(0, \mathbf{p}) = 0$ *for all* $\mathbf{p} \neq \epsilon$. *In addition,*

$$\alpha(t, \mathbf{p}) \;=\; \textstyle\sum_{\mathbf{p}'y \in \mathbf{p}} \Psi^p(t, \mathbf{p}'y)\alpha(t-1, \mathbf{p}'), \text{ for } 1 \leq t \leq T, \tag{4.4.6}$$

where the summation is over all $\mathbf{p}', y$ *such that* $\mathbf{p}' \in \mathbf{p}$.

Proof. By definition of α, we have the base case formula $\alpha(0, \epsilon) = 1$, and $\alpha(0, \mathbf{p}) = 0$ for all $\mathbf{p} \neq \epsilon$.

For the recurrence formula, consider any $\mathbf{z}$ such that $|\mathbf{z}| > 0$ and $\mathbf{z} \in \mathbf{p}$. Let $\mathbf{z} = \mathbf{z}'y$,

and $\mathbf{z}' \in \mathbf{p}'$, then by Proposition 33, $\mathbf{p}'y \in \mathbf{p}$. Hence

$$\begin{aligned}
\alpha(t, \mathbf{p}) &= \textstyle\sum_{|\mathbf{z}|=t, \mathbf{z}\in\mathbf{p}} P(\mathbf{z}) \\
&= \textstyle\sum_{\mathbf{p}'y\in\mathbf{p}} \sum_{|\mathbf{z}'|=t-1, \mathbf{z}'\in\mathbf{p}'} P(\mathbf{z}'y) \\
&= \textstyle\sum_{\mathbf{p}'y\in\mathbf{p}} \sum_{|\mathbf{z}'|=t-1, \mathbf{z}'\in\mathbf{p}'} \Psi^p(t, \mathbf{p}'y) P(\mathbf{z}') \\
&= \textstyle\sum_{\mathbf{p}'y\in\mathbf{p}} \Psi^p(t, \mathbf{p}'y) \sum_{|\mathbf{z}'|=t-1, \mathbf{z}'\in\mathbf{p}'} P(\mathbf{z}') \\
&= \textstyle\sum_{\mathbf{p}'y\in\mathbf{p}} \Psi^p(t, \mathbf{p}'y) \alpha(t-1, \mathbf{p}')
\end{aligned}$$

■

The backward case is similar, but we describe the key steps for the sake of completeness and preciseness. Define suffix score

$$S(\mathbf{y}, t) \stackrel{\text{def}}{=} \exp(\sum_{i=1}^{m} \sum_{t'=t+|\mathbf{z}^i|-1}^{T} \lambda_i f_i(\mathbf{x}, \mathbf{y}, t')). \tag{4.4.7}$$

If $\mathbf{z} = \mathbf{y}_{t:T} = \mathbf{y}'_{t:T}$, then $S(\mathbf{y}, t) = S(\mathbf{y}', t)$, thus we can define $S(\mathbf{z}) = S(\mathbf{y}, t)$. The *backward variable* $\beta(t, \mathbf{s})$ is defined by

$$\beta(t, \mathbf{s}) \stackrel{\text{def}}{=} \sum_{\mathbf{z}:|\mathbf{z}|=T+1-t, \mathbf{z}\in\mathbf{s}} S(\mathbf{z}). \tag{4.4.8}$$

Let $\Psi^s_{\mathbf{x}}(t, \mathbf{s}) \stackrel{\text{def}}{=} \exp(\sum_{\mathbf{z}^i \text{ is a prefix of } \mathbf{s}} \lambda_i g_i(\mathbf{x}, t + |\mathbf{z}^i| - 1))$. by definition, $\beta(T+1, \epsilon) = 1$, and $\beta(T+1, \mathbf{s}) = 0$ for all $\mathbf{s} \neq \epsilon$. The recurrence for β is

$$\beta(t, \mathbf{s}) = \textstyle\sum_{y\mathbf{s}'\in\mathbf{s}} \Psi^s(t, y\mathbf{s}') \beta(t+1, \mathbf{s}') \text{ for } 1 \leq t \leq T. \tag{4.4.9}$$

Once α or β is computed, then Z can be easily obtained:

$$Z = \textstyle\sum_{\mathbf{p}} \alpha(T, \mathbf{p}) = \sum_{\mathbf{s}} \beta(1, \mathbf{s}). \tag{4.4.10}$$

Time Complexity: Assume that each evaluation of the function $g_i(\cdot, \cdot)$ can be performed in unit time for all i. [1] All relevant values of $\Psi^p_{\mathbf{x}}$ that are used can hence be computed in $O(m|\mathcal{Y}||\mathcal{P}|T)$ time. In practice, this is pessimistic, and the computation can be done more quickly. For all following analyses, we assume that Ψ^p has already been computed and stored in an array. Now all values of α can be computed in $\Theta(|\mathcal{Y}||\mathcal{P}|T)$ time. Similar bounds for Ψ^s and β hold.

4.4.2 Viterbi Decoding

As in the case of HMM (Rabiner, 1989), Viterbi decoding (calculating the most likely label sequence) is obtained by replacing the sum operator in the forward backward algorithm with the max operator.

Formally, let

$$\delta(t, \mathbf{p}) \stackrel{\text{def}}{=} \max_{|\mathbf{z}|=t, \mathbf{z} \in \mathbf{p}} P(\mathbf{z}). \tag{4.4.11}$$

By definition, $\delta(0, \epsilon) = 1$, and $\delta(0, \mathbf{p}) = 0$ for all $\mathbf{p} \neq \epsilon$. The recurrence is given by

$$\delta(t, \mathbf{p}) \quad = \quad \max_{\mathbf{p}' y \in \mathbf{p}} \Psi^p(t, \mathbf{p}' y) \delta(t-1, \mathbf{p}'), \text{ for } 1 \leq t \leq T. \tag{4.4.12}$$

We use $\Phi(t, \mathbf{p})$ to record the pair $(\mathbf{p}', y)$ chosen to obtain $\delta(t, \mathbf{p})$,

$$\Phi(t, \mathbf{p}) \quad = \quad \arg \max_{\mathbf{p}' y \in \mathbf{p}} \Psi^p(t, \mathbf{p}' y) \delta(t-1, \mathbf{p}'). \tag{4.4.13}$$

Let $\mathbf{p}^*_T = \arg\max_{\mathbf{p}} \delta(T, \mathbf{p})$, then the most likely path $\mathbf{y}^* = (y^*_1, \ldots, y^*_T)$ has y^*_T as the last label in $\mathbf{p}^*_T$, and the full sequence can be traced backwards using $\Phi(\cdot, \cdot)$ as

[1] This is acceptable in many cases, but not necessarily true. For example, if $\mathbf{x}$ is a sequence of words and we define $g(\mathbf{x}, t)$ as the number of words in $\mathbf{x}$ at most some edit distance away from the t-th word, then computing $g(\mathbf{x}, t)$ takes time linear in the length of the sentence and the length of the t-th word if the standard DP algorithm is used.

follows

$$(\mathbf{p}_t^*, y_t^*) \quad = \quad \Phi(t+1, \mathbf{p}_{t+1}^*), \text{ for } 1 \leq t < T. \tag{4.4.14}$$

Time Complexity: Either Ψ^p or Ψ^s can be used for decoding; hence decoding can be done in $\Theta(|\mathcal{Y}| \min\{|\mathcal{P}|, |\mathcal{S}|\}T)$ time.

4.4.3 Marginals

We need to compute marginals of label sequences and single variables, that is, compute $P(\mathbf{y}_{t-|\mathbf{z}|:t} = \mathbf{z}|\mathbf{x})$ for $\mathbf{z} \in \mathcal{Z} \cup \mathcal{Y}$. Unlike in the traditional HMM, additional care need to be taken regarding features having label patterns that are super or sub sequences of $\mathbf{z}$. We define

$$W(t, \mathbf{z}) \quad = \quad \exp(\textstyle\sum_{(i,j):\mathbf{z}^i \subset_j \mathbf{z}} \lambda_i g_i(\mathbf{x}, t - |\mathbf{z}| + j)).$$

This function computes the sum of all features that may activate strictly within $\mathbf{z}$.

For any $\mathbf{z}$, if $|\mathbf{z}| > 1$, then let $\mathbf{z}^- = \mathbf{z}_{2:|\mathbf{z}|}$ and $\mathbf{z}_- = \mathbf{z}_{1:|\mathbf{z}|-1}$. If $|\mathbf{z}| = 1$, then let $\mathbf{z}^- = \mathbf{z}_- = \mathbf{z}$. If $\mathbf{p}$ ends with $\mathbf{z}_-$ and $\mathbf{s}$ starts with $\mathbf{z}^-$, define $[\mathbf{p}, \mathbf{z}, \mathbf{s}]$ as the sequence $\mathbf{p}_{1:|\mathbf{p}|-|\mathbf{z}_-|}\mathbf{z}\mathbf{s}_{(|\mathbf{z}^-|+1):|\mathbf{s}|}$, and

$$O(t, \mathbf{p}, \mathbf{s}, \mathbf{z}) \quad = \quad \exp(\textstyle\sum_{\mathbf{z} \subseteq \mathbf{z}^k, \mathbf{z}^k \subseteq_i [\mathbf{p},\mathbf{z},\mathbf{s}]} \lambda_k g_k(\mathbf{x}, t - |\mathbf{p}| + i - 1)).$$

$O(t, \mathbf{p}, \mathbf{s}, \mathbf{z})$ counts the contribution of features with their label patterns properly containing $\mathbf{z}$ but within $[\mathbf{p}, \mathbf{z}, \mathbf{s}]$.

Proposition 35. *Let $\mathbf{z} \in \mathcal{Z} \cup \mathcal{Y}$. For any $\mathbf{y}$ such that $\mathbf{y}_{1:t}$ ends with $\mathbf{z}$, there exist unique $\mathbf{p}, \mathbf{s}$ such that $\mathbf{p}$ ends with $\mathbf{z}^-$, $\mathbf{s}$ starts with $\mathbf{z}_-$, $\mathbf{y}_{1:t-1} \in \mathbf{p}$, and $\mathbf{y}_{t-|\mathbf{z}^-|+1:T} \in \mathbf{s}$. In addition, $Z(\mathbf{y}) = \frac{1}{W(t,\mathbf{z})} P(\mathbf{y}, t-1) S(\mathbf{y}, t - |\mathbf{z}^-| + 1) O(t, \mathbf{p}, \mathbf{s}, \mathbf{z})$.*

Multiplying by O counts features that are not counted in $P(\mathbf{y}_{1:t-1}) S(\mathbf{y}_{t-|\mathbf{z}|+2:T})$

while division by W removes features that are double-counted. By Proposition 35, we have

$$P(\mathbf{y}_{t-|\mathbf{z}|+1:t} = \mathbf{z}|\mathbf{x}) = \frac{\sum_{\mathbf{p} \text{ ends with } \mathbf{z}_-,\ \mathbf{s} \text{ starts with } \mathbf{z}^-} \alpha(t-1,\mathbf{p})\beta(t-|\mathbf{z}^-|+1,\mathbf{s})O(t,\mathbf{p},\mathbf{s},\mathbf{z})}{ZW(t,\mathbf{z})}.$$

Time Complexity: Let L denote the maximum order of the features, then both $W(t,\mathbf{z})$ and $O(t,\mathbf{p},\mathbf{s},\mathbf{z})$ can be computed in $O(|\mathbf{p}||\mathbf{s}|) = O(L^2)$ time. Let $c(\mathcal{Z})$ denotes the sum all products $|\mathbf{p}||\mathbf{s}|$ over triplets $(\mathbf{p},\mathbf{z},\mathbf{s})$ satisfying $\mathbf{z} \in \mathcal{Z}$, $\mathbf{p}$ ends with $\mathbf{z}_-$, and $\mathbf{s}$ starts with $\mathbf{z}^-$, then $P(\mathbf{y}_{t-|\mathbf{z}|+1:t} = \mathbf{z}|\mathbf{x})$ can be computed in time $O(c(\mathcal{Z}))$. A simple upper bound for $c(\mathcal{Z})$ is $L^2|\mathcal{P}||\mathcal{S}|$.

4.5 Training

Given a training set $\mathcal{T}$, the model parameters λ_i's can be chosen by maximizing the regularized log-likelihood $\mathcal{L}_\mathcal{T} = \ln \Pi_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} P(\mathbf{y}|\mathbf{x}) - \sum_{i=1}^m \frac{\lambda_i^2}{2\sigma_{reg}^2}$, where σ_{reg} is a parameter that controls the degree of regularization. Note that $\mathcal{L}_\mathcal{T}$ is a $\cap$-convex function of $\lambda_1,\ldots,\lambda_m$. Let $F_i(\mathbf{x},\mathbf{y}) = \sum_t f_i(\mathbf{x},\mathbf{y},t)$, then its maximum is achieved when

$$\frac{\partial \mathcal{L}_\mathcal{T}}{\partial \lambda_i} = \tilde{\mathrm{E}}(F_i) - \mathrm{E}(F_i) - \frac{\lambda_k}{\sigma_{reg}^2} = 0,$$

where

$$\tilde{\mathrm{E}}(F_i) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} F_i(\mathbf{x},\mathbf{y}), \tag{4.5.1}$$

is the empirical average of F_i, and

$$\mathrm{E}(F_i) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \sum_{|\mathbf{y}'|=|\mathbf{x}|} P(\mathbf{y}'|\mathbf{x})F_i(\mathbf{x},\mathbf{y}'), \tag{4.5.2}$$

is the expected value of F_i. Given the gradient and value of $\mathcal{L}_{\mathcal{T}}$, we use the L-BFGS optimization method (Liu and Nocedal, 1989), for maximizing the regularized log-likelihood, as it has been demonstrated to be a superior method for optimizing CRFs (Sha and Pereira, 2003).

The function $\mathcal{L}_{\mathcal{T}}$ can now be computed because we have shown how to compute $Z_{\mathbf{x}}$, and computing the value of $Z_{\mathbf{x}}(\mathbf{y})$ is straightforward, for all $(\mathbf{x}, \mathbf{y}) \in \mathcal{T}$. For the gradient, computing $\tilde{\mathrm{E}}(F_i)$ is straightforward, and $\mathrm{E}(F_i)$ can be computed using marginals computed in previous section:

$$\mathrm{E}(F_i) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} \sum_{t} P(\mathbf{y}'_{t-|\mathbf{z}^i|+1:t} = \mathbf{z}^i|\mathbf{x}) g_i(\mathbf{x}, t). \tag{4.5.3}$$

Time Complexity: Computing the gradient is clearly more time-consuming than $\mathcal{L}_{\mathcal{T}}$, thus we shall just consider the time needed to compute the gradient. Let $l = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} |\mathbf{x}|$. We need to compute at most $|\mathcal{Z}|l$ marginals, thus total time needed to compute all the marginals is $O(L^2|\mathcal{Z}||\mathcal{P}||\mathcal{S}|l)$. Given the marginals, we can compute the gradient in $O(mL)$ time. The running time for each iteration is $O((L^2|\mathcal{Z}||\mathcal{P}||\mathcal{S}|+m)l)$.

4.6 Extensions

We briefly discuss several extensions of the above algorithms.

4.6.1 Generalized Partition Functions

We demonstrate that our algorithm for computing a very general class of quantities which generalizes the partition function, and will thus be called the generalized partition.

In the following, we assume a fixed set of features, and use $\vec{\lambda}$, with or without decorations, to denote a feature weight vector. We consider a fixed observation sequence

$\mathbf{x}$. In previous section we have omitted the weight vector in the notations since only one weight vector is considered. Here we shall explicitly include the weight vector. For example, we write $Z(\mathbf{y})$ as $Z(\mathbf{y}; \vec{\lambda})$. For integers a, b, we define the generalized partition function

$$Z^{a,b}(\vec{\lambda}_1, \vec{\lambda}_2) \stackrel{\text{def}}{=} \sum_{\mathbf{y}} [Z(\mathbf{y}; \vec{\lambda}_1)]^a [\ln Z(\mathbf{y}; \vec{\lambda}_2)]^b \tag{4.6.1}$$

The generalized partition function can be used to compute various important quantities. We first illustrate this, and then give an $O(|b|^3 |\mathcal{Y}| \min(|\mathcal{P}|, |\mathcal{S}|)T)$ time algorithm to compute it.

The usual partition function $Z(\vec{\lambda})$ is just $Z^{1,0}(\vec{\lambda}, \vec{\lambda})$. It is easy to verify the following formula which express entropy, conditional entropy, and expected feature sum in terms of the generalized partition functions, and thus they can be efficiently computed.

(a) (Entropy) Given an observation sequence, the entropy $H(Y|\mathbf{x}, \vec{\lambda})$ for its label sequence distribution as given by a CRF with parameter $\vec{\lambda}$ is

$$-\sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}; \vec{\lambda}) \ln p(\mathbf{y}|\mathbf{x}; \vec{\lambda}) = \ln Z(\vec{\lambda}) - \frac{Z^{1,1}(\vec{\lambda}, \vec{\lambda})}{Z(\vec{\lambda})}. \tag{4.6.2}$$

The entropy can be used as a regularizer for semi-supervised learning (Jiao et al., 2006; Mann and McCallum, 2007), or used for example selection in active learning (Kim et al., 2006). To use it with gradient methods for training, we also show how the gradient of entropy can be computed efficiently. First we have (Mann and McCallum, 2007)

$$\frac{\partial H(Y|\mathbf{x}, \vec{\lambda})}{\lambda_i} \tag{4.6.3}$$

$$= -\sum_{\mathbf{y}} (F_i(\mathbf{x}, \mathbf{y}) - F_i(\mathbf{x})) p(\mathbf{y}|\mathbf{x}, \vec{\lambda}) \ln p(\mathbf{y}|\mathbf{x}, \vec{\lambda}) \tag{4.6.4}$$

$$= -\sum_{\mathbf{y}} F_i(\mathbf{x}, \mathbf{y}) p(\mathbf{y}|\mathbf{x}, \vec{\lambda}) \ln p(\mathbf{y}|\mathbf{x}, \vec{\lambda}) + F_i(\mathbf{x}) H(Y|\mathbf{x}, \vec{\lambda}), \tag{4.6.5}$$

where $F_i(\mathbf{x}) = \mathrm{E}(F_i(\mathbf{x},\mathbf{y})|\mathbf{x},\vec{\lambda})$. We have shown how to compute $F_i(\mathbf{x})$ and $H(Y|\mathbf{x},\vec{\lambda})$, thus the second term can be computed. For the first term, note that

$$\begin{aligned} & -\textstyle\sum_{\mathbf{y}} F_i(\mathbf{x},\mathbf{y}) p(\mathbf{y}|\mathbf{x},\vec{\lambda}) \ln p(\mathbf{y}|\mathbf{x},\vec{\lambda}) && (4.6.6)\\ = & -\textstyle\sum_{\mathbf{y}} \sum_t f_i(\mathbf{x},\mathbf{y},t) p(\mathbf{y}|\mathbf{x},\vec{\lambda}) \ln p(\mathbf{y}|\mathbf{x},\vec{\lambda}) && (4.6.7)\\ = & -\textstyle\sum_t f_i(\mathbf{x},\mathbf{y},t) \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x},\vec{\lambda}) \ln p(\mathbf{y}|\mathbf{x},\vec{\lambda}) && (4.6.8)\\ = & -\textstyle\sum_t g_i(\mathbf{x},t) \sum_{\mathbf{y}\text{ends with }\mathbf{z}_i\text{ at }t} p(\mathbf{y}|\mathbf{x},\vec{\lambda}) \ln p(\mathbf{y}|\mathbf{x},\vec{\lambda}). && (4.6.9) \end{aligned}$$

Thus it suffices to compute all $H(Y|\mathbf{x},\vec{\lambda}, Y$ ends with $\mathbf{z}_i$ at t) for all different $\mathbf{z}_i$ and t values. This can be computed using the same method as for computing the entropy, but restricting the possible values for those $|\mathbf{z}_i|$ labels ending at t.

(b) (KL-divergence) The KL-divergence between two CRFs with $\vec{\lambda}_1$ and $\vec{\lambda}_2$ as the parameters is given by

$$\textstyle\sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x};\vec{\lambda}_1) \ln \frac{p(\mathbf{y}|\mathbf{x};\vec{\lambda}_1)}{p(\mathbf{y}|\mathbf{x};\vec{\lambda}_2)} = \frac{Z^{1,1}(\vec{\lambda}_1,\vec{\lambda}_1) - Z^{1,1}(\vec{\lambda}_1,\vec{\lambda}_2)}{Z(\vec{\lambda}_1)} - [\ln Z(\vec{\lambda}_1) - \ln Z(\vec{\lambda}_2)]. \quad (4.6.10)$$

(c) (Expected feature sum) Let $\vec{\lambda}'$ be the unit vector with the kth component being 1, then

$$\mathrm{E}(F_k(\mathbf{x},\mathbf{y},t;\vec{\lambda})|\mathbf{x}) = Z^{1,1}(\vec{\lambda},\vec{\lambda}')/Z(\vec{\lambda}). \quad (4.6.11)$$

Using the above equation, we can compute the gradient of the loglikelihood function in $O(m|\mathcal{Y}|\min\{|\mathcal{P}|,|\mathcal{S}|\}T)$ time.

We now sketch how to compute the generalized partition function. First, note that $Z^{a,b}(\vec{\lambda}_1,\vec{\lambda}_2) = Z^{1,b}(a\vec{\lambda}_1,\vec{\lambda}2)$, thus it suffices to consider the case when $a = 1$. Define the generalized forward vector

$$\alpha_b(t,\mathbf{p};\vec{\lambda}_1,\vec{\lambda}_2) = \textstyle\sum_{|\mathbf{z}|=t,\mathbf{z}\in\mathbf{p}} P(\mathbf{z};\vec{\lambda}_1)[\ln P(\mathbf{z};\vec{\lambda}_2)]^b. \quad (4.6.12)$$

Then the generalized partition function can be computed as

$$Z^{1,b}(\vec{\lambda}_1, \vec{\lambda}_2) = \sum_{\mathbf{p}} \alpha_b(T, \mathbf{p}; \vec{\lambda}_1, \vec{\lambda}_2). \tag{4.6.13}$$

Hence it suffices to compute the generalized forward vector. For simplicity of notation, we write Ψ^p as Ψ in the following.

Theorem 36. *For $b \geq 0$ and $t > 0$, we have*

$$\alpha_b(t, \mathbf{p}; \vec{\lambda}_1, \vec{\lambda}_2) = \sum_i \binom{b}{i} \sum_{\mathbf{p}' y \in \mathbf{p}} \Psi(t, \mathbf{p}' y; \vec{\lambda}_1)[\ln \Psi(t, \mathbf{p}' y; \vec{\lambda}_2)]^{b-i} \alpha_i(t-1, \mathbf{p}'). \tag{4.6.14}$$

Proof. The proof is similar to that of Theorem 34.

$$\begin{aligned}
& \alpha_b(t, \mathbf{p}; \vec{\lambda}_1, \vec{\lambda}_2) \\
= \; & \sum_{|\mathbf{z}|=t, \mathbf{z} \in \mathbf{p}} P(\mathbf{z}; \vec{\lambda}_1)[\ln P(\mathbf{z}; \vec{\lambda}_2)]^b \\
= \; & \sum_{\mathbf{p}' y \in \mathbf{p}} \sum_{|\mathbf{z}'|=t-1, \mathbf{z}' \in \mathbf{p}'} P(\mathbf{z}' y; \vec{\lambda}_1)[\ln P(\mathbf{z}' y; \vec{\lambda}_2)]^b \\
= \; & \sum_{\mathbf{p}' y \in \mathbf{p}} \sum_{|\mathbf{z}'|=t-1, \mathbf{z}' \in \mathbf{p}'} \Psi(t, \mathbf{p}' y) P(\mathbf{z}')[\ln P(\mathbf{z}'; \vec{\lambda}_2) + \ln \Psi(t, \mathbf{p}' y; \vec{\lambda}_2)]^b \\
= \; & \sum_{\mathbf{p}' y \in \mathbf{p}} \sum_{|\mathbf{z}'|=t-1, \mathbf{z}' \in \mathbf{p}'} \Psi(t, \mathbf{p}' y) P(\mathbf{z}') \sum_i \binom{b}{i} [\ln P(\mathbf{z}'; \vec{\lambda}_2)]^i [\ln \Psi(t, \mathbf{p}' y; \vec{\lambda}_2)]^{b-i} \\
= \; & \sum_i \binom{b}{i} \sum_{\mathbf{p}' y \in \mathbf{p}} \sum_{|\mathbf{z}'|=t-1, \mathbf{z}' \in \mathbf{p}'} \Psi(t, \mathbf{p}' y) [\ln \Psi(t, \mathbf{p}' y; \vec{\lambda}_2)]^{b-i} P(\mathbf{z}') [\ln P(\mathbf{z}'; \vec{\lambda}_2)]^i \\
= \; & \sum_i \binom{b}{i} \sum_{\mathbf{p}' y \in \mathbf{p}} \Psi(t, \mathbf{p}' y) [\ln \Psi(t, \mathbf{p}' y; \vec{\lambda}_2)]^{b-i} \sum_{|\mathbf{z}'|=t-1, \mathbf{z}' \in \mathbf{p}'} P(\mathbf{z}') [\ln P(\mathbf{z}'; \vec{\lambda}_2)]^i \\
= \; & \sum_i \binom{b}{i} \sum_{\mathbf{p}' y \in \mathbf{p}} \Psi(t, \mathbf{p}' y; \vec{\lambda}_1) [\ln \Psi(t, \mathbf{p}' y; \vec{\lambda}_2)]^{b-i} \alpha_i(t-1, \mathbf{p}'; \vec{\lambda}_2).
\end{aligned}$$

■

Take $b = 0$, and $\vec{\lambda}_1 = \vec{\lambda}_2$, then we obtain the recurrence formula in Theorem 34. Using the above recurrence, assuming all Ψ values can be retrieved in unit time, we can compute α_0, α_1, α_2, ..., α_b successively in a total of $O([1^2 + 2^2 + \ldots + (b+1)^2]|\mathcal{P}||\mathcal{Y}|T) = O(b^3|\mathcal{P}||\mathcal{Y}|T)$ time. We can similarly define the backward version to

achieve $O(b^3 \min\{|\mathcal{P}|, |\mathcal{S}|\}\mathcal{Y}T)$ time.

4.6.2 Semi-Markov features

For semi-Markov CRFs using zeroth and first order features, and high-order semi-Markov features of the form

$$f(\mathbf{x}, \mathbf{y}, t) = \mathrm{I}(\mathbf{y}_{1:t} \text{ has a suffix matching } p, \text{while } \mathbf{y}_{1:t+1} \text{ does not}),$$

where p is a label pattern such as $a+b+c+$ (here $a+$ is the regular expression denoting that there is at least one a), can be efficiently trained and decoded. See (Nguyen et al., 2011) for more details.

4.6.3 Incorporating constraints

We can modify the forward and backward algorithm to compute more general partition functions, by setting U appropriately in the following modification of Eq. 4.4.6

$$\alpha(t, \mathbf{p}) \;\; = \;\; \textstyle\sum_{\mathbf{p}'y \in \mathbf{p}} \Psi^p(t, \mathbf{p}'y)\alpha(t-1, \mathbf{p}')U(\mathbf{p}'y, t). \tag{4.6.15}$$

To illustrate, suppose $\mathbf{x}$ is only labeled at certain positions, with those unlabeled positions marked by $\emptyset$, and let $\mathbf{y}$ denote this incomplete label sequence. To compute the partition function $\sum_{|\mathbf{z}| = |\mathbf{x}| \text{ and } \mathbf{z} \text{ agrees with } \mathbf{y} \text{ on labeled positions}} Z(\mathbf{z})$, we just need to set $U(\mathbf{x}, \mathbf{p}'y, t) = \mathrm{I}(\mathbf{y}_{t:t} = \emptyset \text{ or } \mathbf{y}_{t:t} = y)$. This can be used for performing training on partially labeled data, and for computing marginals. Note that this method also provides a method to compute marginals.

Similar modification can be done on the Viterbi algorithm for decoding under various types of constraints.

4.7 Experiments

We first use a synthetic dataset to study conditions under which high-order features can potentially be useful, then we use a handwritten character recognition dataset to demonstrate that incorporating simple high-order features can lead to impressive performance improvement on a real problem. [2]

4.7.1 Labeling High-order Markov Chains

We use high-order Markov chains to generate synthetic data to investigate conditions under which high-order features are useful.

We randomly generate an order k Markov model with n states $s_1, \ldots, s_n$ as follows. To increase pattern sparsity, we allow at most r randomly chosen possible next state given the previous k states. This limits the number of possible label sequences in each length $k+1$ segment from n^{k+1} to $n^k r$. The conditional probabilities of these r next states are generated by randomly selecting a vector from uniform distribution over $[0,1]^r$ and normalizing it. Each state s_i generates an observation $(a_1, \ldots, a_m)$ such that a_j follows a Gaussian distribution with mean μ_{ij} and standard deviation σ. Each $\mu_{i,j}$ is independently randomly generated from the uniform distribution over $[0,1]$. In the experiments, we set $n = 5$, $r = 2$ and $m = 3$.

The standard deviation, σ, has an important role in determining the characteristics of the data generated by this Markov model. If σ is very small as compared to most μ_{ij}'s, then using the observations alone as features is likely to be good enough to obtain a good classifier of the states; the label correlations becomes less important for classification. However, if σ is large, then it is difficult to distinguish the states based on the observations alone and the label correlations, particularly those captured by higher order features are likely to be helpful. In short, the standard deviation, σ, is

[2] The results presented are better than those in (Ye et al., 2009) due to a bug in the decoding algorithm.

used to control how much information the observations reveal about the states.

We use the current, previous and next observations, rather than just the current observation as features, exploiting the conditional probability modeling strength of CRFs. For higher order features, we simply use all indicator features that appeared in the training data up to a maximum order. We considered the case $k = 2$ and $k = 3$, and varied σ and the maximum order. The training set and test set each contains 500 sequences of length 20; each sequence was initialized with a random sequence of length k and generated using the randomly generated order k Markov model. Training was done by maximizing the regularized log likelihood with regularization parameter $\sigma_{\text{reg}} = 1$. The experimental results are shown in Figure 4.1.

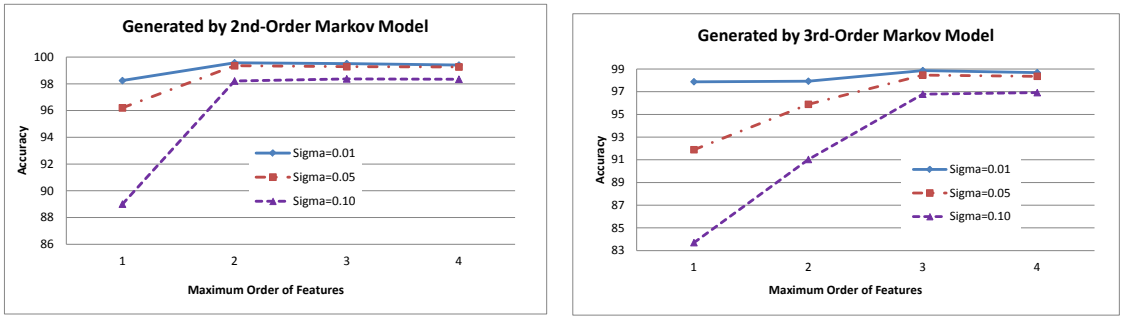


Figure 4.1: Accuracy as a function of maximum order on the synthetic data set.

Figure 4.1 shows that the high-order indicator features are useful in this case. In particular, we can see that it is beneficial to increase the order of the high-order features when the underlying model has longer distance correlations. As expected, increasing the order of the features beyond the order of the underlying model is not helpful. The results also suggest that in general, if the observations are closely coupled with the states (in the sense that different states correspond to very different observations), then feature engineering on the observations is generally enough to perform well, and it is less important to use high-order features to capture label correlations. On the other hand, when such coupling is not clear, it becomes important to capture the label

correlations, and high-order features can be useful.

4.7.2 Handwriting Recognition

We used the handwriting recognition data set from (Taskar et al., 2003), consisting of around 6100 handwritten words with an average length of around 8 characters. The data was originally collected by Kassel. (1995) from around 150 human subjects. The words were segmented into characters, and each character was converted into an image of 16 by 8 binary pixels. In this labeling problem, each x_i is the image of a character, and each y_i is a lower-case letter. The experimental setup is the same as that used in (Taskar et al., 2003): the data set was divided into 10 folds with each fold having approximately 600 training and 5500 test examples and the zero-th order features for a character are the pixel values.

For higher order features, we again used all indicator features that appeared in the training data up to a maximum order. The average accuracy over the 10 folds are shown in Figure 4.2, where strong improvements are observed as the maximum order increases. Figure 4.2 also shows the total training time and the running time per iteration of the L-BFGS algorithm (which requires computation of the gradient and value of the function at each iteration). The running time appears to grow no more than linearly with the maximum order of the features for this data set. Note that our results shows a significant improvement (around 8%) over previously reported best, which is around 88% (Taskar et al., 2003; Qian et al., 2009).

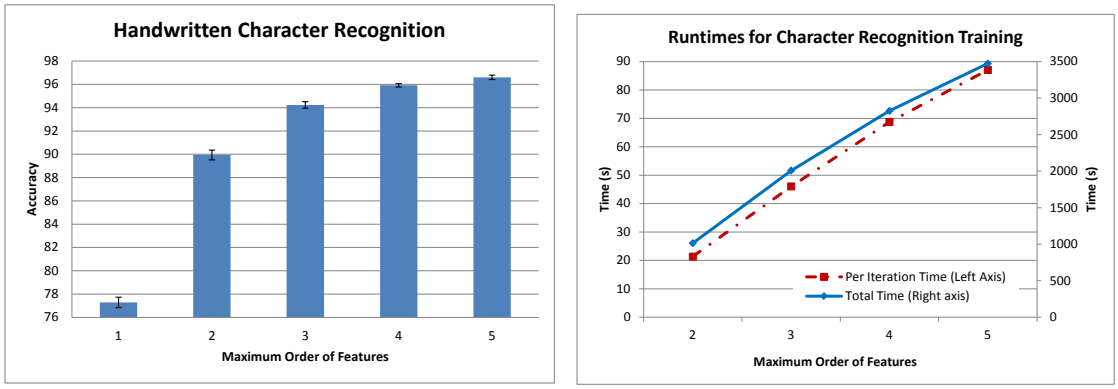


Figure 4.2: Accuracy (left) and running time (right) as a function of maximum order for the handwriting recognition data set.

4.8 Discussion

Our algorithms rely on the sparsity of the patterns that the high-order potential functions will activate, and once we know these patterns, the time complexity can be estimated exactly to see whether the algorithms will run in feasible time. In our experiments, we used indicator features of all label patterns that appear in the training data. For real applications, if the pattern sparsity assumption is not satisfied, but certain patterns do not appear frequently enough and are not really important, then it is useful to see how we can select a subset of features with few distinct label patterns automatically. One possible approach would be to use boosting type methods (Dietterich et al., 2004) to sequentially select useful features.

An alternative approach to feature selection is to use all possible features and maximize the margin of the solution instead. Generalization error bounds (Taskar et al., 2003) show that it is possible to obtain good generalization with a relatively small training set size despite having a very large number of features if the margin is large. This indicates that feature selection may not be critical in some cases.

It should also be possible to use kernels within the approach here. On the handwritten character problem, Taskar et al. (2003) reports substantial improvement in performance with the use of kernels. Use of kernels together with high-order features may lead to further improvements. However, we note that the advantage of the higher order features may become less substantial as the observations become more powerful in distinguishing the classes. Whether the use of higher order features together with kernels brings substantial improvement in performance is likely to be problem dependent.

Although the technique for handling sparse high-order features can be easily generalized to handle simple sparse high-order semi-Markov features, it is not clear how these algorithms can be extended to handle general high-order semi-Markov features in which the dependency between the labels and the observations are coupled. An example of such feature is *the first segment contains the word Peter, and the second segment contains the word palace*. For a feature like this, we can refine the states to memorize partial information like whether the previous segment contains the word Peter. However, this method is ad hoc and requires tailoring the algorithm each time a different type of dependency is introduced, for example, *all the segments shares a common word*. In addition, the number of states can grow quickly as the number of features grow. This is an avenue that should be explored for further research on exploiting sparsity.

Chapter 5

Sparse Factorial CRFs for Sequence Multi-Labeling

We continue our discussion on sparse CRFs in this chapter by presenting sparse FCRFs for capturing both temporal and co-temporal constraints. This can be useful in sequence multi-labeling problems, such as multiple activity recognition, as pointed out in previous chapter.

Section 5.1 first motivates FCRF as a model useful for capturing both temporal and co-temporal dependencies, and outlines the contribution in our sparse FCRFs . Section 5.2 reviews some related works. Section 5.3 describe sparse FCRFs. Section 5.4 presents our exact efficient inference algorithms for sparse FCRFs. Section 5.5 describes the training algorithm used. Section 5.6 demonstrates that our sparse FCRFs can effectively capture and exploit both temporal and co-temporal dependencies, on synthetic datasets and an activity recognition dataset, as compared to varous baselines.

5.1 Capturing Temporal and Co-temporal Dependencies

Sequence multi-labeling, or labeling an observation sequence with multiple correlated label sequences, arise in various domains. For example, sequence multi-labeling is required for labeling sentences with noun-phrase chunking tags and parts-of-speech (POS) tags in natural language processing (Sutton et al., 2007), and for labeling video frames with multiple concepts in multimedia (Li et al., 2010).

Sequence multi-labeling can be done by building a multi-label classifier for each observation, or building an independent sequence labeler for each individual label sequence. However, a multi-label classifier does not fully exploit temporal correlations, and independent sequence labelers do not fully exploit co-temporal (spatial, in the case of the video annotation) dependencies. An approach to capture both temporal and co-temporal dependencies is to solve the sequence labeling problems one after another by feeding the output of an earlier stage as the input to the next one. This is commonly done in the pipelining approach used in natural language processing, but errors in the earlier stages can cascade down the pipeline.

Factorial conditional random field (FCRF) (Sutton et al., 2007) is a joint conditional probabilistic model which has the advantage of capturing both temporal and co-temporal dependencies without suffering from cascading errors. However, inference becomes computationally intractable in a FCRF as the number of label sequences increases. In this chapter, we consider sparse factorial conditional random field (SFCRF): the class of FCRFs with co-temporal potential functions which are constant except at a small number of co-temporal label patterns. Such models are beneficial when we have useful information to incorporate at only a few possible patterns at each temporal location but do not wish to rule out the possibility of the sequence being labeled with other labels at those locations. In particular, we can use the class of SFCRF to

combine the output of a few algorithms, where the combined outputs of the algorithms may not necessarily contain the true co-temporal label pattern. We give a polynomial time exact inference algorithm for computing marginals for SFCRFs, and show that it can be scaled to problems with many label sequences. Our algorithm is a forward-backward algorithm, but in contrast to the usual forward-backward algorithms, we do not maintain a fixed-sized table for the sufficient statistics (the information contained in the part of the sequence that has been processed). Instead, a sum product representation, which grows linearly with the part of the sequence that has been processed, is used for each entry in the usual tabular representation.

We evaluate SFCRFs on a synthetic dataset and an activity recognition dataset, against two methods: a multi-label classifier that uses only co-temporal information, and independent sequence labelers. We use MaxEnt classifiers and CRFs as the underlying models, with different decoding methods. On the synthetic dataset, SFCRF is competitive with or better than the best of the two methods under different noise conditions or with distribution shift. On the activity recognition dataset, SFCRFs obtains significantly better performance than the two approaches. We further compare the running time of the new algorithm with the running time of an approximate inference algorithm on the activity recognition dataset. Interestingly, the new exact inference algorithm is significantly faster than the tree reparameterized loopy belief propagation algorithm (TRP/BP) (Wainwright et al., 2001), both when TRP/BP is used on the SFCRF model and when it is used on a FCRF model with pairwise potentials.

5.2 Related Works

We discuss related works on modeling for sequence multi-labeling and works on exploiting sparsity in potential functions for efficient inference and learning.

Sequence multi-labeling models are generally extensions of sequence labeling mod-

els. Factorial hidden Markov model (FHMM) (Ghahramani and Jordan, 1997) is an extension of hidden Markov models (HMM) (Rabiner, 1989), and have been applied to joint noun-phrasing chunking and POS tagging (Duh, 2005). However, as a generative model, capturing arbitrary dependencies between labels and observations is difficult in FHMM. FCRFs (Sutton et al., 2007) extends linear-chain conditional random fields (CRFs) (Lafferty et al., 2001), and has the advantage of being able to incorporate arbitrary dependencies between labels and observations without significantly increasing the inference and learning cost. SVMs (Tsochantaridis et al., 2004) have also beeen extended for sequence multi-labeling (Li et al., 2010). All the above-mentioned works noted the computational difficulty of handling complex co-temporal dependencies between the labels, and used approximate inference algorithms, such as tree reparametrized loopy belief propagation algorithm (TRP/BP) Wainwright et al. (2001) (used in (Sutton et al., 2007)), graph-cut algorithms for optimizing binary MRFs (Rother et al., 2007) (used in (Li et al., 2010)).

Note that within the context of this work, sparsity means that the logarithm of the co-temporal potential functions are mostly zero in their domains, and should be differentiated from works on parameter sparsity (such as works on ℓ_1 regularization) which seeks to eliminate the effects of entire potential functions. It should also be differentiated with the simple method of restricting the possible labels at each time step to a small number of possibilities. This method gives an efficient algorithm but gives zero probability to labels that are not in the allowed set. In contrast, SFCRF assigns non-zero probability to every possible label sequence and is more robust, as we show in the experiments.

Sparse potential functions, as used in this paper, have been previously exploited for improving the efficiency of inference. Qian et al. (2009) gave a forward-backward algorithm for a class sparse high-order CRFs by performing inference on groups of states and transitions with same potential function. The time complexity of their

algorithm is not polynomial in the worst case. Ye et al. (2009) considered high-order CRFs which are in a subclass of those considered in (Qian et al., 2009), and derived polynomial time forward-backward inference algorithms by grouping forward variables and backward variables using the locations at which the potential functions are active. Nguyen et al. (2011) extended the algorithm in (Ye et al., 2009) to give a polynomial time inference algorithms for high-order semi-Markov CRFs. In this paper, we use a different approach towards exploiting sparse co-temporal potential functions. At a high-level, our algorithm is the same as the standard forward-backward algorithm. The novelty lies in the use of a sum-product representation for the forward and backward values, instead of the use of a fixed sized table. Given this representation, we are able to efficiently compute all the marginals required for inference and learning.

5.3 Sparse Factorial CRFs

Let $\mathbf{x} = (x_1, \ldots, x_T)$ be an observation sequence, and let $\mathbf{y}_i = (y_{i1}, y_{i2}, \ldots, y_{iT})$ be the i-th label sequence. We shall use $\mathbf{y}_{i:j}$ to denote the sequences $\mathbf{y}_i, \mathbf{y}_{i+1}, \ldots \mathbf{y}_j$. For general CRFs with K label sequences, we have features of the form $f_i(\mathbf{x}, \mathbf{y}_{1:K}, t)$, with the weight for f_i being λ_i. The weight for $\mathbf{y}_{1:K}$ is defined as

$$Z(\mathbf{x}, \mathbf{y}_{1:K}) = \exp(\textstyle\sum_i \sum_t \lambda_i f_i(\mathbf{x}, \mathbf{y}_{1:K}, t))$$

Then the CRF associated with the label sequences is

$$\mathrm{P}_{\vec{\lambda}}(\mathbf{y}_{1:K}|\mathbf{x}) = Z(\mathbf{x}, \mathbf{y}_{1:K})/Z(\mathbf{x})$$

where $Z(\mathbf{x}) = \sum_{\mathbf{y}'_{1:K}} Z(\mathbf{x}, \mathbf{y}'_{1:K})$.

For FCRFs, the features are of the following three possible forms: zeroth-order feature $f_{ki}(\mathbf{x}, y_{kt}, t)$, first-order feature $g_{ki}(\mathbf{x}, y_{k,t-1:t}, t)$, and co-temporal feature $q_i(\mathbf{x}, y_{\cdot t}, t)$,

where $y_{k,a:b} = (y_{k,a}, y_{k,a+1}, \ldots, y_{k,b})$, and $y_{\cdot t} = (y_{1t}, \ldots, y_{Kt})$.

We can group the factors in $Z(\mathbf{x}, \mathbf{y}_{1:K})$ based on the label variables they depend on, and write it in the form

$$Z(\mathbf{x}, \mathbf{y}_{1:K}) = \prod_{k,t} \phi_k^{(t)}(y_{kt}) \prod_{k,t} \psi_k^{(t)}(y_{k,t-1}, y_{kt}) \prod_t \theta^{(t)}(y_{\cdot t}), \tag{5.3.1}$$

where

$$\begin{aligned} \phi_k^{(t)}(y_{k,t}) &= \exp(\textstyle\sum_i \lambda_{ki} f_{ki}(\mathbf{x}, y_{kt}, t)), \\ \psi_k^{(t)}(y_{k,t-1}, y_{k,t}) &= \exp(\textstyle\sum_i \lambda'_{ki} g_{ki}(\mathbf{x}, y_{k,t-1:t}, t), \\ \theta^{(t)}(y_{\cdot t}) &= \exp(\textstyle\sum_i \mu_i q_i(\mathbf{x}, y_{\cdot,t}, t)). \end{aligned}$$

Here λ_{ki}, λ'_{ki} and μ_i are the weights for f_{ki}, g_{ki} and q_i respectively. $\phi^{(t)}$, $\psi^{(t)}$ and $\theta^{(t)}$ are called the potentials associated with the their argument variables. For a given sequence, $\mathbf{x}$ is constant and we omit it from the potential notations.

Our sparse FCRFs make use of only sparse co-temporal features: at each time step, all the q_i's are non-zero only for a small number of co-temporal patterns. This implies that, for any t, the log potential function $\log \theta^{(t)}(\cdot)$ is non-zero for only a small number of values in its domain.

We describe three types of sparse co-temporal features, assuming a given attribute $s(\mathbf{x}, t)$. First, consider a small pre-defined set $P = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ of co-temporal patterns. These patterns can be used to construct a set of sparse pattern features

$$q_i(\mathbf{x}, \mathbf{z}, t) = \mathrm{I}\mathbf{z} = \mathbf{q}_i s(\mathbf{x}, t) \tag{5.3.2}$$

where $I(\cdot)$ is the indicator function. A large weight for q_i indicates that $\mathbf{q}_i$ is more likely to occur at each time slice.

Next, we consider the case where we have several prediction algorithms $\{\mathcal{A}\}$, and

each $\mathcal{A}$ outputs a small set $P_t(\mathcal{A})$ of likely co-temporal patterns at time slice t (such as the most likely few co-temporal patterns predicted by a probabilistic co-temporal pattern classifier $\mathcal{A}$). We can define features of the following form

$$q_{\mathcal{A}}(\mathbf{x}, \mathbf{z}, t) = \mathrm{I}\mathbf{z} \in P_t(\mathcal{A}) s(\mathbf{x}, t). \tag{5.3.3}$$

In this case, $P_t(\mathcal{A})$ can be time-varying and is used to exploit the predictions of the algorithms. A large weight for $q_{\mathcal{A}}$ indicates that patterns in $P_t(\mathcal{A})$ are more likely to occur at time slice t.

The above two types of features can be combined to yield features of the following form

$$q_{i,\mathcal{A}}(\mathbf{x}, \mathbf{z}, t) = \mathrm{I}\mathbf{z} = \mathbf{q}_i \mathrm{I}\mathbf{q}_i \in P_t(\mathcal{A}) s(\mathbf{x}, t). \tag{5.3.4}$$

This may be useful, for example, when we want to restrict the patterns from the algorithm, $\mathcal{A}$, to also belong to the patterns observed in the training set.

In the following, for simplicity of presentation, assume all label variables take values from the same finite set $\mathcal{Y}$. Let $n = |\mathcal{Y}|$. We also assume there are m co-temporal patterns $\mathbf{q}_1, \ldots, \mathbf{q}_m$ from $\mathcal{Y}^K$ such that for each q_i, $q_i(\mathbf{x}, y_{\cdot t}, t) \neq 0$ only if $y_{\cdot t}$ equals to some $\mathbf{q}_j$. Our algorithms can be easily generalized to the case when the set of patterns is time-varying. The requirement for our algorithms to work efficiently is only that a small set of patterns can be active at each time slice.

5.4 Inference

We consider computing marginals of the form $\mathrm{P}(y_{it}|\mathbf{x})$, $\mathrm{P}(y_{i,t-1}, y_{i,t}|\mathbf{x})$, or $\mathrm{P}(y_{\cdot t}|\mathbf{x})$. First note that direct generalization of the forward-backward algorithm (see Rabiner, 1989) needs to deal with n^K states, and the forward and backward variables need $O(n^{2K}T)$ time to compute. Using the computed forward and backward values, computing a single

$\mathrm{P}(y_{\cdot t}|\mathbf{x})$ requires $O(1)$ time (given the partition function), computing a single $\mathrm{P}(y_{it}|\mathbf{x})$ requires $O(n^K)$ time, and computing a single $\mathrm{P}(y_{i,t-1}, y_{i,t}|\mathbf{x})$ requires $O(n^{2K})$ time. All the marginals of the above forms can be computed in a more efficient manner in a total time of $O(n^{2K}KT)$. We present exact polynomial-time algorithms for computing any single marginal of the above forms for SFCRF, and show in Section 5.5 that this leads to a polynomial time learning algorithm.

Using the notation in Eq. 5.3.1, if $\mathbf{y}'_1, \ldots, \mathbf{y}'_K$ are sequences of the same length used to label the first few observations, we define their prefix weight $Z^p(\mathbf{x}, \mathbf{y}'_{1:K})$ as

$$Z^p(\mathbf{x}, \mathbf{y}'_{1:K}) = \prod_{i,t\leq|\mathbf{y}'_1|} \phi_i^{(t)}(y'_{i,t}) \prod_{i,t\leq|\mathbf{y}'_1|} \psi_i^{(t)}(y'_{i,t-1}, y'_{i,t}) \prod_{t\leq|\mathbf{y}'_1|} \theta^{(t)}(\mathbf{y}'_{\cdot t}), \tag{5.4.1}$$

where $|\cdot|$ denotes the length of a sequence.

Let $\mathbf{z} = (z_1, \ldots, z_K) \in \mathcal{Y}^K$. Define the forward variable

$$\alpha(t, \mathbf{z}) = \sum_{|\mathbf{y}_1|=\ldots=|\mathbf{y}_K|=t, y_{\cdot t}=\mathbf{z}} Z^p(\mathbf{x}, \mathbf{y}_{1:K}) \tag{5.4.2}$$

We have $Z(\mathbf{x}) = \sum_{\mathbf{z}} \alpha(T, \mathbf{z})$.

Lemma 37. *$\alpha(t, \mathbf{z})$ can be represented in sum-product form as*

$$\alpha(t, \mathbf{z}) = \sum_{i=1}^{1+mt} u_{1,i}^{(t)}(z_1) \ldots u_{ki}^{(t)}(z_k). \tag{5.4.3}$$

It takes $O(mKT)$ time to compute a single value of $\alpha(t, \mathbf{z})$. In addition, given the sum-product representation of $\alpha(t, \mathbf{z})$, it takes $O((n^2 + m)mtK)$ time to compute that of $\alpha(t + 1, \mathbf{z})$. Hence, all $\alpha(t, \cdot)$'s can be computed in $O((n^2 + m)mT^2K)$ time.

Proof.(Lemma 37) We give a constructive inductive proof below. The case $t = 0$ is

trivial by setting

$$\begin{aligned} u_{j,1}^{(0)}(z_j) &= 1, \\ \alpha(0, \mathbf{z}) &= u_{1,1}^{(0)}(z_1) \dots u_{K1}^{(0)}(z_K), \end{aligned}$$

Now we demonstrate how to construct a sum-product representation for $\alpha(t+1, \mathbf{z})$ using a sum-product representation for $\alpha(t, \mathbf{z})$. This is done by first multiplying in ϕ and ψ potentials, then multiplying in the θ potentials. For $1 \leq i \leq 1 + mt$, we first take the factors from the chains into account by defining

$$u_{k,i}^{(t+1)}(z_k) = \textstyle\sum_{z'_k} u_{k,i}^{(t)}(z'_k) \psi_k^{(t+1)}(z'_k, z_k) \phi_k^{(t)}(z_k)$$

These factors can be computed in $O(n^2 mtK)$ time. We define the following forward variables, leaving the co-temporal potentials unaccounted for,

$$\alpha'(t+1, \mathbf{z}) = \textstyle\sum_{i=1}^{1+mt} u_{1,i}^{(t+1)}(z_1) \dots u_{Ki}^{(t+1)}(z_K)$$

We have $\alpha(t+1, \mathbf{z}) = \alpha'(t+1, \mathbf{z})$ for all $\mathbf{z}$ not equal to any $\mathbf{q}_i$. It is easy to verify that

$$\alpha(t+1, \mathbf{z}) = \alpha'(t+1, \mathbf{z}) + \alpha'(t+1, \mathbf{z})(\theta^{(t+1)}(\mathbf{z}) - 1)$$

The second term vanishes for all $\mathbf{z}$ except at m positions $\mathbf{q}_1, \dots, \mathbf{q}_m$. For $i = 1$ to m, we can construct $u_{j,1+mt+i}^{(t+1)}$ such that $u_{1,1+mt+i}^{(t+1)}(z_1) \dots u_{K,1+mt+i}^{(t+1)}(z_K)$ is equal to $\alpha'(t+1, \mathbf{q}_i)(\theta^{(t+1)}(\mathbf{q}_i) - 1)$ when $\mathbf{z}$ is equal to $\mathbf{q}_i$ and vanishes otherwise, then

$$\alpha(t+1, \mathbf{z}) = \textstyle\sum_{i=1}^{1+m(t+1)} u_{1,i}^{(t+1)}(z_1) \dots u_{K,i}^{(t+1)}(z_K)$$

For the m additional terms, clearly each value of $\alpha(t, \mathbf{z})$ can be computed in $O(mtK)$ time, and creating each new product term takes $O(nK)$ time. Thus introducing these

terms take $O(m^2tK + mnK)$ time. Hence the total time to construct the sum-product representation for $\alpha(t+1, \mathbf{z})$ given that of $\alpha(t, \mathbf{z})$ is $O(n^2mtK + m^2tK + mnK) = O((n^2+m)mtK)$. The total time to compute all sum-product representations is thus $O(\sum_t (n^2+m)mtK) = O((n^2+m)mT^2K)$. ■

Note that if the chains are independent, then the time required for computing the forward variables is $O(n^2TK)$. Thus the blowup in time complexity of the above algorithm is by a factor of $\frac{n^2+m}{n^2}mT$.

Using the above sum-product representation, we can compute the partition function in polynomial-time, and thus the likelihood as well.

Proposition 38. *$Z(\mathbf{x})$ can be computed in $O((n^2+m)mT^2K)$ time.*

Proof. Given the sum-product representation in Proposition 37, the partition function can be computed in $O(nmTK)$ time using variable elimination as follows:

$$\begin{aligned} Z(\mathbf{x}) &= \textstyle\sum_{\mathbf{z}} \alpha(T, \mathbf{z}) \\ &= \textstyle\sum_{i=1}^{1+mT} (\sum_{z_1} u_{1,i}^{(T)}(z_1)) \ldots (\sum_{z_K} u_{K,i}^{(T)}(z_K)) \end{aligned}$$

Since the sum-product representation takes $O((n^2+m)mT^2K)$ to compute, the total time is $O((n^2+m)mT^2K)$. ■

To compute the marginals, we first define the backward variable $\beta(t, \mathbf{z})$ as the sum of suffix weights for all combinations of suffix label sequences starting from position t, with the label variables at position t being $\mathbf{z} = (z_1, \ldots, z_k)$. We can similarly show that $\beta(t, \mathbf{z})$ can be represented in a sum-product form as

$$\beta(t, \mathbf{z}) = \textstyle\sum_{i=1}^{1+m(T-t+1)} v_{1,i}^{(t)}(z_1) \ldots v_{ki}^{(t)}(z_k). \tag{5.4.4}$$

Given the forward and backward variables, we can obtain the sum of weights of sequences agreeing at the same time slice or two consecutive time slices.

Lemma 39. *(a) Let $S(t, \mathbf{z})$ be the sum of the weight of all label sequences $\mathbf{y}_1, \ldots, \mathbf{y}_K$ for $\mathbf{x}$ such that the time t slice is $\mathbf{z}$, then*

$$S(t, \mathbf{z}) = \frac{\alpha(t, \mathbf{z})\beta(t, \mathbf{z})}{\phi^{(t)}(z_1) \ldots \phi^{(t)}(z_k)} \left(\frac{1}{\theta^{(t)}(\mathbf{z})} - 1 \right) + \sum_i \sum_j \prod_k \frac{u_{k,i}^{(t)}(z_k) v_{k,j}^{(t)}(z_k)}{\phi^{(t)}(z_k)}$$

Note that the first term vanishes for all $(z_1, \ldots, z_k)$ except at m positions $\mathbf{q}_1, \ldots, \mathbf{q}_m$.

(b) Let $\mathbf{z}' = (z'_1, \ldots, z'_K)$. Let $S(t, \mathbf{z}', \mathbf{z})$ be the sum of the weight of all label sequences $\mathbf{y}_1, \ldots, \mathbf{y}_K$ for $\mathbf{x}$ such that the time $t-1$ and t slices are $\mathbf{z}'$ and $\mathbf{z}$ respectively, then

$$S(t, \mathbf{z}', \mathbf{z}) = \sum_i \sum_j \prod_k u_{k,i}^{(t-1)}(z'_k) v_{k,j}^{(t)}(z_k) \psi^{(t)}(z'_k, z_k)$$

Using variable elimination on $S(t, \mathbf{z})$ and $S(t, \mathbf{z}', \mathbf{z})$, and normalizing with the partition function, we can compute marginals for one variable, and marginals for two variables respectively. The marginals for a time slice can be directly obtained by normalizing $S(t, \mathbf{z})$ with the partition function.

Proposition 40. *Given the sum-product representations of the forward and backward vectors,*

(a) $\{\mathrm{P}(y_{kt}|\mathbf{x}) : 1 \le k \le K, 1 \le t \le T\}$ *can be computed in $O(nm^2T^3K)$ time.*

(b) $\{\mathrm{P}(y_{k,t-1}y_{k,t}|\mathbf{x}) : 1 \le k \le K, 1 \le t \le T\}$ *can be computed in $O(n^2m^2T^3K)$ time.*

(c) $\{\mathrm{P}(y_{\cdot t} = \mathbf{q}_i|\mathbf{x}) : 1 \le i \le m, 1 \le t \le T\}$ *can be computed in $O(m^3T^3K)$ time.*

If the sum-product representations are not given, then the total time needed to find all the above marginals is $O(\max(n^2, m)m^2T^3K)$.

Unfortunately, the above sum-product representation does not easily carry over to an algorithm for Viterbi decoding – applying the max operator to a sum-product

representation seems to destroy the sum-product representation at the next step. We outline how an algorithm for computing the marginals can be used to approximate Viterbi decoding by using the standard temperature argument.

Proposition 41. *(Wainwright and Jordan, 2008, p.196) Let $Y_1, \ldots, Y_l$ be an arbitrary set of discrete random variables, and suppose $P(Y_1, \ldots, Y_l|t, \vec{\lambda}) \propto e^{\vec{\lambda} \cdot T(Y_1,\ldots,Y_l)/t}$, where t is called a temperature parameter. Denote the most likely (breaking ties arbitrarily) configuration for a single variable at temperature t as*

$$y'_{i,t} = \arg\max_{y_i} P(y_i|t, \vec{\lambda}). \tag{5.4.5}$$

Then when $t > 0$ is sufficiently small, $y'_{i,t}$ is part of a most likely configuration of the variables.

Proposition 41 can be used to compute a most likely configuration by sequentially setting the value of a single variable and re-doing the inference to find new marginals at a sufficiently small temperature. If the most likely variable configuration is unique, then the most likely configuration for a single variable will also be unique and sequential setting of the variables is not required.

5.5 Training

Model parameters yielding high likelihood on training data indicates good fit, but overfitting may occur. Regularized maximum likelihood is more often used for training. Formally, let $\mathcal{T}$ denote the training set, $\mathcal{T}_x$ denote the observation sequences in $\mathcal{T}$, then $\vec{\lambda} = (\lambda_i)$ is chosen to maximize

$$\mathcal{L}(\vec{\lambda}) = \ln \textstyle\prod_{\mathbf{x} \in \mathcal{T}_x} P(\mathbf{y}_{1:K}|\mathbf{x}) - \sum_i \frac{\lambda_i^2}{2\sigma_{reg}^2},$$

where the term $-\sum_i \frac{\lambda_i^2}{2\sigma_{reg}^2}$ is the Gaussian regularizer with σ_{reg} controlling the degree of regularization. Note that $\mathcal{L}(\vec{\lambda})$ is a $\cap$-convex function of $\vec{\lambda}$, and its maximum is achieved when

$$\frac{\partial \mathcal{L}}{\partial \lambda_i} = \tilde{\mathrm{E}}(f_i) - \mathrm{E}(f_i) - \frac{\lambda_k}{\sigma_{reg}^2} = 0$$

where

$$\begin{aligned} \tilde{\mathrm{E}}(f_i) &= \sum_{\mathbf{x} \in \mathcal{T}_x} \sum_t f_i(\mathbf{x}, \mathbf{y}_{1:K}, t) \\ \mathrm{E}(f_i) &= \sum_{\mathbf{x} \in \mathcal{T}_x} \sum_{\mathbf{y}'_{1:K}} P(\mathbf{y}'_{1:K}|\mathbf{x}) \sum_t f_i(\mathbf{x}, \mathbf{y}'_{1:K}, t) \end{aligned}$$

Here, $\tilde{\mathrm{E}}(f_i)$ is the empirical sum of the feature f_i in the observed data, and $\mathrm{E}(f_i)$ is the expected sum of f_i according to the model specified by $\vec{\lambda}$. While there are various algorithms for maximizing the regularized log-likelihood of CRFs, the L-BFGS algorithm (Liu and Nocedal, 1989) has been shown to demonstrate good performance (Sha and Pereira, 2003). We omit details on the L-BFGS algorithm, and simply treat it as a black box which requires computation of the gradient and value of $\mathcal{L}(\vec{\lambda})$.

The regularized log-likelihood $\mathcal{L}(\vec{\lambda})$ can be easily computed because computing $Z(\mathbf{x}, \mathbf{y}_{1:K})$ is straightforward and $Z(\mathbf{x})$ can be computed using Proposition 38. For the gradient, computing $\tilde{\mathrm{E}}(f_i)$ is straightforward, and $\mathrm{E}(f_i)$ can be computed using marginals computed in previous section:

$$\mathrm{E}(f_{ki}) = \sum_{\mathbf{x} \in \mathcal{T}_x} \sum_t \sum_{y'_{kt}} P(y'_{kt}|\mathbf{x}) f_{ki}(\mathbf{x}, y'_{kt}, t) \tag{5.5.1}$$

$$\mathrm{E}(g_{ki}) = \sum_{\mathbf{x} \in \mathcal{T}_x} \sum_t \sum_{y'_{k,t-1}, y'_{k,t}} P(y'_{k,t-1} y'_{k,t}|\mathbf{x}) g_{ki}(\mathbf{x}, y'_{k,t-1:t}, t) \tag{5.5.2}$$

$$\mathrm{E}(q_i) = \sum_{\mathbf{x} \in \mathcal{T}_x} \sum_t \sum_{y'_{\cdot t}} P(y'_{\cdot t}|\mathbf{x}) q_i(\mathbf{x}, \mathbf{y}'_{\cdot t}, t) \tag{5.5.3}$$

Note that for the last equation, $y'_{\cdot t}$ ranges over $\mathbf{q}_1, \ldots \mathbf{q}_m$, because for other values of $y'_{\cdot t}$, the feature takes value 0.

5.6 Experiments

In this section, we evaluate SFCRFs on a synthetic dataset and an activity recognition dataset. We compare SFCRFs performance with various baseline algorithms to demonstrate that SFCRFs can be an effective model, and we show that with sufficient sparsity, our exact inference algorithm is much faster than a loopy belief propagation algorithm.

The following baseline models are used. ME_{disj} trains a MaxEnt classifier for each individual label and predicts the most like label. It exploits neither the temporal nor co-temporal correlations. CRF_{disj} trains a linear-chain CRF for each label sequence and predicts the most like sequences. This baseline exploits the temporal correlations but not the co-temporal ones. ME_{pat} trains a MaxEnt classifier using all the observed co-temporal patterns in the training set as a label set, where each label indicates a co-temporal pattern – in this case, if there are m observed patterns in the training set, it trains on a label set of size m. This exploits the co-temporal correlation but not the temporal correlations. CRF_{pat} use all observed co-temporal labels as states in the CRF. It exploits both temporal and co-temporal correlations but assigns zero probability to sequences that contains co-temporal patterns that are never seen in the training set.

For ME_{pat} and CRF_{pat}, decoding can be done in various ways. CRF^{ml}_{pat} predicts the most likely pattern at each time slice, CRF^{vtb}_{pat} predicts the most likely pattern sequence, and CRF^{md}_{pat} first marginalizes the distribution of the patterns to obtain the distributions for each label, then predicts the most likely value of each label. To be precise, given a co-temporal pattern distribution $P(\mathbf{y} = \mathbf{p}_i)$'s, we compute $P(y_k = 1)$ as $\sum_{\mathbf{p}_{i,k}=1} P(\mathbf{y} = \mathbf{p}_i)$, and then predict most likely y_k's. In this way, it is able to predict pattern that has never been seen in the training set, even though CRF_{pat} assigns zero probability to those patterns. We use the same marginalization technique for ME^{ml}_{pat} and ME^{md}_{pat}.

5.6.1 Synthetic Datasets

We first use a synthetic dataset to study how the time complexity of SFCRF scales as K increases, then use two synthetic dataset to demonstrate that SFCRFs are able to exploit both temporal and co-temporal dependencies in a way better than the baselines.

A. Scaling of time complexity We generate a synthetic dataset with small number of co-temporal patterns. The patterns consists of those K binary patterns with exactly one 1. A random Markov transition model with K states, where each state correspond to one of these binary patterns is used to generate the label sequences. Only a dummy observation with value 1 is observed. We generate 60 training and 140 test sequences of length 10. We used Gaussian regularizer for all models with $\sigma^2_{reg} = 1$.

For FCRF, we consider a sparse FCRF and a dense one. For the SFCRF, the features used are: (a) For each chain, include all the observations from the chains as zero-th order features and all label transitions as first order features at current time slice; (b) The co-temporal features are the sparse pattern features defined in Eq. 5.3.2, with P consisting of the K co-temporal patterns, and $s(x, t)$ be the $3K$ observed real random variables together with one attribute having constant value 1. The dense FCRF used all the 2^K co-temporal patterns instead of just those K seen patterns. Inference in the dense FCRF is done using the exponential-time forward-backward algorithm discussed in Section 5.4.

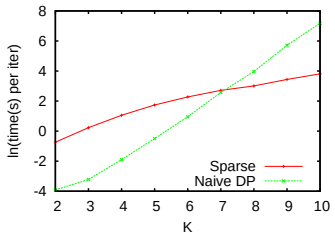


Figure 5.1: Logarithm of the per-iteration time (s) in L-BFGS for our algorithm and the naive algorithm.

Both SFCRF and the dense FCRF predict the most likely label at each position. We first observe the time required for one iteration of computing the regularized log-likelihood and its gradient, using our new algorithm and the exponential-time forward-backward algorithm. The timings are averages from several runs and are plotted in log

scale for K from 2 to 10 in Figure 5.1. As expected, our algorithm shows a log-shape curve and the naive algorithm shows a linear curve. The SFCRF algorithm starts to become faster when around 7 chains are used, indicating that the constants in the running time are practically reasonable.

B. Capturing co-temporal information to deal with noisy observations We used the same Markov transition model as above, with 10 chains but only 4 states corresponding to 4 label patterns. The 4 patterns have their first 8 bits randomly chosen, but the last two bits are constructed as the xor bit of the first 8 bits and its complement. For the observation model, a label $y \in \{0, 1\}$ in the k-th sequence uses a Gaussian $N(4(y, y, y), 4kI_3)$ to generate the observations. Thus observations for chains with higher indices are noisier, and the observations by themselves do not provide sufficient information on the last two bits. Thus for this dataset, co-temporal dependencies are expected to be useful, and independent classifiers or taggers that depend only on temporal correlations are not expected to perform as well. The label accuracies for the algorithms are shown in Table 5.1. Approaches exploiting co-temporal dependencies perform better, and SFCRF is able to perform slightly better than algorithms that use only co-temporal information.

Table 5.1: Accuracies of the baseline algorithms and SFCRF using noisy observation.

	ME_{disj}	ME^{ml}_{pat}	ME^{md}_{pat}	CRF_{disj}	CRF^{ml}_{pat}	CRF^{vtb}_{pat}	CRF^{md}_{pat}	SFCRF
Acc	88.42	90.40	90.18	89.20	93.20	92.95	93.28	93.87

C. Using temporal information to predict unseen patterns We used the same Markov transition model as in B, but with only two states corresponding to two randomly chosen complementary patterns. Furthermore, each label y use $N(4(y, y, y), I_3)$ to generate observations giving strong information about the labels. In the test set, the two patterns in the training set are replaced with two different randomly chosen complementary patterns. As the labels have changed, the co-temporal information be-

comes useless in the test set. However, the temporal correlation learned in the training set remains useful, as the underlying 2-state Markov transition model and observation model given the label remains the same. In this case, we expect the independent chains to perform well and the co-temporal pattern based algorithms to perform badly. The results, as shown in Table 5.2, indicate that this is indeed the case. SFCRF is able to perform almost as well as CRF_{disj} indicating that it is able to back off to the independent CRFs when encountering novel patterns not seen in training.

Table 5.2: Accuracies of the baseline algorithms and SFCRF on test set with different label patterns.

	ME_{disj}	ME^{ml}_{pat}	ME^{md}_{pat}	CRF_{disj}	CRF^{ml}_{pat}	CRF^{vtb}_{pat}	CRF^{md}_{pat}	SFCRF
Acc	73.59	59.63	59.63	99.41	59.63	59.63	59.63	98.26

5.6.2 Multiple Activities Recognition

We evaluated SFCRFs on the activity recognition data set from (Patterson et al., 2005). The dataset consists of 10 sequences of roughly half an hour of activities. At each time step, a person can be performing one or several activities among 11 activities (for example, making soft boiled eggs while making tea). The observations are the RFID labels of the objects being touched, and each observation is labeled with one or several activities that the subject is performing. We first combined the data into 1 data set, then collapsed identical consecutive observation-label pairs into one. Then we broke the long activity sequence into sequences, breaking the sequence whenever all current activities stop or the sequence length is 30 – note that since only first-order transitions are captured in the usual models, breaking long sequences into short ones will not cause significant loss of information in general. A total of 30% of the data is used for training, and the remaining 70% used for testing.

We considered the following features: (a) Observations within windows of size 3 as

zero-th order features, and all label transitions as first order features for each chain; (b) Co-temporal features using the patterns predicted by any of some chosen algorithms using the observations within windows of size 3 as input, that is, we used features of the type in Eq. 5.3.3, which are defined as

$$q_{\mathcal{A}}(\mathbf{x}, \mathbf{y}_{1:K}, t) = \mathrm{I}y_{\cdot t} = \mathbf{p}(\mathcal{A}, \mathbf{x}, t)s(\mathbf{x}, t), \tag{5.6.1}$$

where $\mathbf{p}(\mathcal{A}, \mathbf{x}, t)$ is the pattern predicted by algorithm $\mathcal{A}$ at time t, and $s(\mathbf{x}, t)$ is an observation within windows of size 3, or a constant 1. We give results for using a single $q_{\mathcal{A}}$ feature for each baseline algorithm $\mathcal{A}$, and using all $q_{\mathcal{A}}$ features. We performed 10-fold cross-validation to select the best regularization parameter from $\sigma^2_{reg} = 0.1, 1, 10$.

We measured the following performance measures for our algorithm: Acc, the label accuracy; #eval, the number of function and gradient evaluations used by L-BFGS; T_{tot}, the total running time for training in seconds; and T_{eval}, the running time per evaluation. We also compared our exact inference algorithm with tree reparameterized loopy belief propagation algorithm (TRP/BP) (Wainwright et al., 2001). The implementation in GRMM (Sutton, 2006) was used. Both GRMM and our code are written in Java, and the experiments are run on Linux OS using single threads. Since TRP/BP runs faster with pairwise potentials, we also consider a FCRF *Pair*, which does not have the co-temporal pattern features, but has all the pairwise potentials for labels within the same time slice. TRP/BP is a randomized algorithm, thus we run it using five randomly selected seeds, and for each of the performance measures, we report the average over those five runs.

The results are shown in Table 5.3. For each baseline algorithm A, except CRF_{disj}, the FCRF with sparse co-temporal features from A outperforms A significantly according to McNemar's test (Dietterich, 1998). The failure to improve CRF_{disj} is not surprising as CRF_{disj} does not really provide new co-temporal information for SFCRF.

Table 5.3: Accuracies of the evaluated algorithms on the activity recognition dataset.

	ME_{disj}	ME^{ml}_{pat}	ME^{md}_{pat}	CRF_{disj}	CRF^{ml}_{pat}	CRF^{vtb}_{pat}	CRF^{md}_{pat}	All	Pair
					Baseline				
Acc	91.71	91.30	91.82	92.41	91.84	92.84	91.35	-	-
					FCRF				
Acc	93.02	93.17	93.28	92.11	93.75	93.41	92.97	94.13	-
#eval	29	28	31	29	30	32	29	29	-
$T_{tot}(s)$	158	164	171	149	169	165	155	1335	-
$T_{eval}(s)$	5	6	6	5	6	5	5	45	-
					GRMM				
Acc	93.13	93.05	93.06	92.19	93.24	93.09	93.03	92.82	92.07
#eval	48.4	46.8	33.8	46	56.4	49.2	57.2	66.8	55.4
$T_{tot}(s)$	31289	28275	26715	37654	45221	45046	37957	51275	6537
$T_{eval}(s)$	652	596	789	821	774	904	671	759	119

In addition, using all q_A features yields the best performance of 94.13%. This shows sparse co-temporal features can be very useful, and our algorithms make integrating the outputs of several algorithms efficiently doable. We further investigate the reason SFCRF is able to outperform the baselines. We found that the training set contains 30 distinct patterns and the test set contains 33 distinct patterns but only 15 of the patterns are in both sets. Hence, it seems likely that SFCRF is able to exploit the patterns when they have been seen before, and back off to the linear chains on patterns that have never been seen in the training set.

In terms of accuracies, our exact algorithm and the approximate algorithm in GRMM are very close, indicating that the approximation is mostly sufficient for the problem. However, our algorithm is faster than the approximate algorithm when there is enough sparsity, even though it is doing exact inference. It is even faster than *Pair*.

5.7 Extensions

5.7.1 Incorporating Pattern Transitions

A feature which is informative in our synthetic dataset but not captured in our SFCRF is the transition between co-temporal patterns. Our method can be extended to handle such sparse correlations without changing the asymptotic complexity in inference.

We now describe how to capture transitions between co-temporal patterns when the co-temporal patterns are sparse. Consider features of the form $r_i(\mathbf{x}, y_{\cdot,t-1:t}, t)$. The corresponding potential functions are defined by

$$\gamma^{(t)}(y_{\cdot,t-1}, y_{\cdot,t}) = \textstyle\prod_i r_i(\mathbf{x}, y_{\cdot,t-1}, y_{\cdot,t}, t) \tag{5.7.1}$$

We say r_i's are sparse if $\gamma^{(t)}$ is nonzero for a small number of co-temporal pattern pairs. For simplicity of presentation, we shall assume $\gamma^{(t)}(y_{\cdot,t-1}, y_{\cdot,t}) \neq 0$ iff both $y_{\cdot,t-1}$ and $y_{\cdot,t} \in P = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$. As in the construction of the sum-product representation for α in Proposition 37, the auxiliary function $\alpha'(t+1, \mathbf{z})$ is obtained by incorporating the potentials from the chains into $\alpha(t, \mathbf{z})$. Our sparsity assumption again guarantees that $\alpha(t+1, \mathbf{z}) = \alpha'(t+1, \mathbf{z})$ for all $\mathbf{z} \notin P$. We then handle those $\mathbf{z} \in P$ by adding two correction terms

$$\alpha(t+1, \mathbf{z}) \;=\; \alpha'(t+1, \mathbf{z}) + C_1(\mathbf{z}) + C_2(\mathbf{z}),$$

where

$$\begin{aligned} C_1(\mathbf{z}) &= \alpha'(t+1, \mathbf{z})(\theta^{(t+1)}(\mathbf{z}) - 1), \\ C_2(\mathbf{z}) &= \textstyle\sum_{\mathbf{z}' \in P} \alpha(t, \mathbf{z}')\varphi^{(t+1)}(\mathbf{z})\psi^{(t+1)}(\mathbf{z}', \mathbf{z})\theta^{(t+1)}(\mathbf{z})(\gamma^{(t+1)}(\mathbf{z}', \mathbf{z}) - 1). \end{aligned}$$

Here $\varphi^{(t+1)}(\mathbf{z}) = \prod_k \varphi^{(t+1)}(z_k)$ and $\psi^{(t+1)}(\mathbf{z}', \mathbf{z}) = \prod_k \psi^{(t+1)}(z'_k, z_k)$. All required $C_1(\mathbf{z})$

and $C_2(\mathbf{z})$ values, and the associated m new terms can again can be computed in $O(m^2tK)$ time. Thus Proposition 37 still holds.

The backward variable can be similarly computed. Given the forward and backward variables, $S(t, \mathbf{z})$ can still be computed using the same formula in Proposition 39, but the formula for $S(t, \mathbf{z}', \mathbf{z})$ need to be changed to

$$\begin{aligned} S'(t, \mathbf{z}', \mathbf{z}) &= \textstyle\sum_i \sum_j \prod_k u_{k,i}^{(t-1)}(z'_k) v_{k,j}^{(t)}(z_k) \psi^{(t)}(z'_k, z_k), \\ S(t, \mathbf{z}', \mathbf{z}) &= S'(t, \mathbf{z}', \mathbf{z}) + (\gamma^{(t)}(\mathbf{z}', \mathbf{z}) - 1) S'(t, \mathbf{z}', \mathbf{z}). \end{aligned}$$

Without the first-order co-temporal pattern features, $S(t, \mathbf{z}', \mathbf{z})$ has at least $(1+m)(1+m(T-1))$ product terms. Now we will need to create m^2 additional product terms in the sum-product representation of $S(t, \mathbf{z}', \mathbf{z})$. This does not change the asymptotic complexity of evaluating $S(t, \mathbf{z}', \mathbf{z})$. Proposition 40 still holds.

5.7.2 Combining Sparse High-order and Co-temporal Features

We can extend sparse FCRFs by incorporating high-order features in the chains. We shall call such CRFs as high-order FCRFs (HFCRFs). The following paragraphs briefly describe how inference can be done for HFCRFs.

The key idea is to replace the states for each chain as the high-order feature states as in previous chapter. For simplicity of presentation, we assume the chains share the same pattern set $\mathcal{Z}$, and the maximum order is L. Thus they have the same forward-state set $\mathcal{P}$. Let $\mathbf{z} = (z_1, \ldots, z_k)$, where each z_i is an element in $\mathcal{P}$. Define the forward variable

$$\bar{\alpha}(t, \mathbf{z}) = \sum_{|\mathbf{y}_1|=\ldots=|\mathbf{y}_k|=t, y_{1t} \in z_1, \ldots, y_{Kt} \in z_K} Z^p(\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_k). \tag{5.7.2}$$

We have $Z(\mathbf{x}) = \sum_{\mathbf{z}} \alpha(T, \mathbf{z})$. For $\mathbf{z} = (z_1, \ldots, z_k)$, let $\mathbf{e}(\mathbf{z})$ denote the vector $(y'_1, \ldots, y'_n)$ where y'_i is the last label of z_i. $\mathbf{z}$ is said to be consistent with $\mathbf{q}_i$ if $\mathbf{e}(\mathbf{z}) = \mathbf{q}_i$. Let M

denote the number $\mathbf{z}$'s consistent with some $\mathbf{q}_i$. The Ψ^p factors (see Section 4.4) for the k-th chain will be denoted by Ψ^p_k.

Proposition 42. *$\bar{\alpha}(t, \mathbf{z})$ can be represented in sum-product form as*

$$\bar{\alpha}(t, \mathbf{z}) = \sum_{i=1}^{1+mt} u^{(t)}_{1,i}(z_1) \dots u^{(t)}_{ki}(z_k). \tag{5.7.3}$$

In addition, assuming all Ψ factors have already been evaluated, then given the sum-product representation of $\alpha(t, \mathbf{z})$, it takes $O(n|\mathcal{P}|MtK)$ time to compute that of $\alpha(t+1, \mathbf{z})$. Hence, all $\alpha(t, \cdot)$'s can be computed in $O(n|\mathcal{P}|MT^2K)$ time.

Proof. We focus on the recursive case. Similar to the the case without high-order features, we first add in the potentials for the chains, then create new terms to handle the θ potentials.

Assume the following sum-product representation for $\alpha(t, \mathbf{z})$ is given

$$\bar{\alpha}(t, \mathbf{z}) = \sum_{i=1}^{1+mt} u^{(t)}_{1,i}(z_1) \dots u^{(t)}_{ki}(z_k).$$

For $1 \leq i \leq 1+mt$, we first take the factors from the chains into account by defining

$$u^{(t+1)}_{k,i}(z_k) = \sum_{z'_k y \in z_k} u^{(t)}_{k,i}(z'_k) \Psi^p_k(t+1, z'_k y).$$

Now we define the following forward variables, leaving the co-temporal potentials unaccounted first,

$$\bar{\alpha}'(t+1, \mathbf{z}) = \sum_{i=1}^{1+mt} u^{(t+1)}_{1,i}(z_1) \dots u^{(t+1)}_{Ki}(z_K).$$

We have $\bar{\alpha}(t+1, \mathbf{z}) = \bar{\alpha}'(t+1, \mathbf{z})$ for all $\mathbf{z}$ not consistent with any $\mathbf{q}_i$. Hence

$$\bar{\alpha}(t+1, \mathbf{z}) = \bar{\alpha}'(t+1, \mathbf{z}) + \bar{\alpha}'(t+1, \mathbf{z})(\theta^{(t+1)}(\mathbf{e}(\mathbf{z})) - 1).$$

Introducing a new term for each $\mathbf{z}$ consisting with some $\mathbf{q}_i$ completes our argument. The time-complexity bound follows easily from the above argument. ■

To compute the marginals, we use the method in Section 4.6 to constrain the labels and evaluate the sum of the weights of sequences consistent with the constraint. Normalizing the sums by the partition function gives us the marginals. For example, to evaluate the marginal $P(\mathbf{y}_{\cdot t} = \mathbf{q}_i|\mathbf{x})$, we only allow the candidate labels at time slice t to be the same as $\mathbf{q}_i$ in the above algorithm. Since we need to evaluate $T(K|\mathcal{Z}| + m)$ marginal values, and running the above algorithm once takes $O(n|\mathcal{P}|MT^2K)$ time, the total time to compute all the marginals is $O((K|\mathcal{Z}| + m)n|\mathcal{P}|MT^3K))$.

It should be noted that M can be exponential in K. For example, if there is a co-temporal pattern $\mathbf{q}_i$ such that for each chain k, the number of forward states ending with $\mathbf{q}_i(k)$ is at least 2, then the number of $\mathbf{z}$ consistent with $\mathbf{q}_i$ is at least 2^K already. Nevertheless, the above algorithm can still be applied if only a few chains use high-order features.

5.8 Discussion

We have demonstrated that SFCRFs can be used to capture useful temporal and co-temporal dependencies together. Our experimental results shows that provided with one or several baseline algorithm(s), SFCRFs can be used to obtain signficantly better performance over them, thus serving as a principled and exact way to combine the outputs of several learning algorithms. Currently we deal with long sequences by chunking them. Although this may be justified because most models only include up to first-order dependencies, it will be interesting to have algorithms that handle the original sequences. Our algorithms can be extended to handle high-order features described in Chapter 4 from a constant number of chains in polynomial time as well.

Current works on sparse features use very different techniques (Qian et al., 2009; Ye et al., 2009; Nguyen et al., 2011), and a general exact method for exploiting sparse potential functions for inference will be interesting.

Chapter 6

Optimizing F-measures

Classical classification theory and algorithms are mainly concerned with maximizing accuracy. However, accuracy is inadequate in various settings and alternative performance measures have been proposed and popularized. For example, when the dataset is imbalanced (i.e. some classes are rare in comparison to other classes), then predicting the most common classes will often result in high accuracy. In this case, one class of commonly used utility functions are the F-measures, which measure the performance of a classifier in terms of its ability to obtain both high recall (recover most of the instances in the rare classes) and high precision (instances in the common classes predicted to be in the rare classes are mostly truly rare). Another popular performance measure for classifiers is the AUC (Area under the ROC Curve) score (Bradley, 1997).

These popular measures share a characteristic which distinguishes them from accuracy: They cannot be expressed as sums of independent contributions from the instances, while accuracy can. We say that such utility or loss functions are non-decomposable, and performance measures like accuracy are said to be decomposable. Classical theory of classification (Vapnik, 1998) cannot be applied for non-decomposable performance measures. It is no longer clear whether algorithms like ERM or its regularized version are still consistent. In addition, non-decomposability

also poses new algorithmic challenges in learning and inference. For example, ERM is generally hard, and decision-theoretic optimal prediction cannot be computed separately for each instance.

We do not have general theory and algorithms for learning and inference with non-decomposable performance measures yet. In this chapter, we formalize the notion of non-decomposability, and provide some general results. However, we shall only explore the simplest setting where training and test data are i.i.d. instances generated by the same distribution, and we focus on learning and inference with F-measures, and presents theoretical justifications and connections between two learning paradigms for optimizing F-measures: the empirical utility maximization (EUM) approach learns a classifier having optimal performance on training data, while the decision-theoretic approach (DTA) learns a probabilistic model and then predicts labels with maximum expected F-measure. The results presented here describe and extend our previous work on optimizing F-measures (Ye et al., 2012).

In Section 6.1, we review previous works on learning to optimize F-measures and categorize them into the EUM approach and the DTA approach. We then describe the differences between these two approaches.

In Section 6.2, we formalize the notion of non-decomposability and demonstrate that several popular performance measures are non-decomposable. We then establish a consistency result for empirical maximization of F-measures, together with bounds on the rate of convergence. This provides some insights into the factors affecting the convergence rate in EUM. In particular, our bounds suggest that rare classes require more data for performance guarantee, which is consistent with our intuition. We then show that thresholding the true conditional distribution on a large i.i.d. test set can perform as well as the best instance classifier, justifying the popular hybrid approach of learning a conditional distribution followed by learning a threshold. We also show that an EUM-optimal classifier and a DTA-optimal classifier are asymptotically equivalent

if the probability measure for any set of instances with the same conditional probability of being relevant is negligible.

In Section 6.3, we give a new $O(n^2)$ time algorithm for computing optimal predictions, assuming independence of labels. Our algorithm can compute optimal predictions on tens of thousand instances within seconds, significantly faster than previous algorithms which require hours or more. [1]

In Section 6.4, we compare EUM and DTA on synthetic and real datasets. Our theoretical results are useful in explaining the experimental results. Empirically, EUM seems more robust against model misspecification, but given a good model, DTA seems better for handling rare classes on small datasets and a common scenario of domain adaptation.

6.1 Two Learning Paradigms

F-measures (van Rijsbergen, 1974) or F-scores have been commonly used in tasks in which it is important to retrieve elements belonging to a particular class correctly without including too many elements of other classes. F-measures are usually preferred to accuracies as standard performance measures in information retrieval (Manning et al., 2009), particularly, when relevant items are rare. They are also popular in information extraction tasks such as named entity recognition (Tjong Kim Sang and De Meulder, 2003) where most of the elements do not belong to a named class, and in multi-label classification (Dembczynski et al., 2011).

Various methods have been proposed for optimizing F-measures. They fall into two paradigms. The empirical utility maximization (EUM) approach learns a classifier having optimal F-measure on the training data. Optimizing the F-measure directly is often difficult as the F-measure is non-convex. Thus approximation methods are often used instead. Joachims (2005) gave an efficient algorithm for maximizing a convex

[1] Code available at http://www.comp.nus.edu.sg/∼yenan/.

lower bound of F-measures for support vector machines, and showed it worked well on text classification. Jansche (2005) gave an efficient algorithm to maximize a non-convex approximation to F-measures using logistic regression models, and showed it works well on a text summarization problem. A simpler method is to optimize the F-measure in two stages: First learn a score function using standard methods such as logistic regression or support vector machines, then select a threshold for the score function to maximize the empirical F-measure. Though simple, this method has been found to be effective and is commonly applied, for example, in text categorization (Yang, 2001).

The decision-theoretic approach (DTA), advocated by Lewis (1995), estimates a probability model first, and then computes the optimal predictions (in the sense of having highest expected F-measure) according to the model. This method has not been commonly applied for F-measures, possibly due to the high computational complexity of existing algorithms for the prediction step. Assuming the independence of labels, Lewis showed that, in the optimal prediction, the probabilities of being positive for irrelevant items are not more than those for relevant items. He also gave a bound for expected F-measures, which can be computed in $O(n)$ time, but can be very loose. Based on Lewis's characterization, Chai (2005) gave an $O(n^3)$ time algorithm to compute optimal predictions, and he gave empirical demonstration for the effectiveness of DTA. Apparently unaware of Chai's work, Jansche (2007) solved the same problem in $O(n^4)$ time. For the general case when the labels are not necessarily independent, Dembczynski et al. (2011) gave an $O(n^3)$ time algorithm given $n^2 + 1$ parameters of the label distribution, but the parameters can be expensive to compute. They also showed that the independence assumption can lead to bad performance in the worst case, but on the practical datasets used in their experiments, methods assuming the independence assumption are at least as good as those not assuming independence.

We have only discussed works on binary classification. There are also algorithms for

optimizing F-measures for tasks with structured output (Tsochantaridis et al., 2005; Suzuki et al., 2006; Daumé et al., 2009) and multilabel tasks (Fan and Lin, 2007; Zhang et al., 2010; Petterson and Caetano, 2010).

Optimality in EUM and DTA are different. EUM considers only instance classifiers (functions mapping instances to labels), and roughly speaking, an optimal classifier is an instance classifier having highest F-measure on a very large test set among all instance classifiers. On the other hand, DTA considers set classifiers (functions mapping sets of instances to sets of labels), and an optimal classifier in DTA is a set classifier having maximum expected F-measure among all set classifiers. Optimality in these two approaches are also achieved differently using different learning objectives. Unless otherwise stated, optimal classifiers refer to EUM-optimal classifiers, and optimal predictions refer to predictions by DTA-optimal classifiers.

6.2 Theoretical Analysis

Let X and Y denote the input and output random variables. We assume there is a fixed but unknown distribution $P(X, Y)$ that generates i.i.d. (X, Y) pairs during training and testing. We use X and Y to denote their domains as well. In the following, unless otherwise stated, $Y = \{0, 1\}$, with 0 for the negative or irrelevant class and 1 for the positive or relevant class. $\mathrm{I}(\cdot)$ is the indicator function.

Let $D_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a set of n (possibly non-i.i.d.) examples, and let $\mathbf{x}$ and $\mathbf{y}$ denote $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ respectively. If the predicted labels are $\mathbf{s} = (s_1, \ldots, s_n)$, then *precision* $p(\mathbf{s}, \mathbf{y})$ is the number of true positives over the number of predicted positives, and *recall* $r(\mathbf{s}, \mathbf{y})$ is the number of true positives over the number of positives. *F_β-measure* (van Rijsbergen, 1974) $F_\beta(\mathbf{s}, \mathbf{y})$ is a weighted harmonic mean

of precision and recall. Formally,

$$F_\beta = (1+\beta^2)/(\beta^2/r(\mathbf{s},\mathbf{y}) + p(\mathbf{s},\mathbf{y})), \tag{6.2.1}$$

where the precision $p(\mathbf{s},\mathbf{y})$ and the recall $r(\mathbf{s},\mathbf{y})$ can be expressed as

$$p(\mathbf{s},\mathbf{y}) = \textstyle\sum_i s_i y_i / \sum_i s_i, \tag{6.2.2}$$

$$r(\mathbf{s},\mathbf{y}) = \textstyle\sum_i s_i y_i / \sum_i y_i. \tag{6.2.3}$$

Thus, an equivalent expression for F_β is

$$F_\beta(\mathbf{s},\mathbf{y}) = \frac{(1+\beta^2)\sum_i s_i y_i}{\beta^2 \sum_i y_i + \sum_i s_i}. \tag{6.2.4}$$

It is easy to verify that F_0 is the precision and F_∞ is the recall. F_1 is most frequently used in practice. Henceforth, we assume $\beta \in (0,\infty)$.

6.2.1 Non-decomposability

We first formalize the notion of non-decomposability, then demonstrate that F-measures and AUC are non-decomposable. In addition, we show that a performance measure, normalized discounted cumulative gain (NDCG) (Järvelin and Kekäläinen, 2002), which is popular in information retrieval, is non-decomposable. In this section, the labels and predicted values are not necessarily binary.

Definition 43. A utility function is a nonnegative real-valued function $U(\mathbf{s},\mathbf{y})$, and U is said to be decomposable if there exist a $g : \mathbb{N} \to \mathbf{R}$ and a non-negative real-valued function u on $Y \times Y$ such that for any $\mathbf{s},\mathbf{y}$

$$U(\mathbf{s},\mathbf{y}) = \frac{1}{g(n)} \sum_i u(s_i, y_i). \tag{6.2.5}$$

If a utility function is decomposable, then

$$\frac{g(n)}{n}U(\mathbf{s},\mathbf{y}) = \frac{1}{n}\sum_i u(s_i, y_i), \tag{6.2.6}$$

thus we can study its properties by applying classical statistical learning theory described in Section 2.3. But as we shall demonstrate below, there are interesting non-decomposable utility functions, and thus their properties cannot be obtained by direct application of classical statistical learning theory.

Proposition 44. *F_β is non-decomposable.*

Proof. Assume F_β is decomposable, then there exists g and u such that

$$F_\beta(\mathbf{s},\mathbf{y}) = \frac{1}{g(n)}\sum_i u(s_i, y_i).$$

Consider the following quantities

$$\begin{aligned}
U_1 &= F_\beta((1,1,1),(1,1,1)) = 1,\\
U_2 &= F_\beta((0,1,1),(1,1,1)) = 2(1+\beta^2)/(3\beta^2+2),\\
U_3 &= F_\beta((1,1,0),(1,1,0)) = 1,\\
U_4 &= F_\beta((0,1,0),(1,1,0)) = (1+\beta^2)/(2\beta^2+1).
\end{aligned}$$

Decomposability implies that

$$U_1 - U_2 = (u(1,1) - u(0,1))/g(3) = U_3 - U_4.$$

However, we have

$$U_1 - U_2 = \beta^2/(3\beta^2+2) < \beta^2/(2\beta^2+1) = U_3 - U_4,$$

a contradiction. Hence F_β is non-decomposable. ■

Various binary classifiers are obtained by thresholding a score function, that is, a function mapping each instance to a real number. For example, for probabilistic classifiers such as naive Bayes or logistic regression, the score for an instance is its probability of being positive, and a threshold of 0.5 is often used to maximize expected accuracy. Another example is the support vector machines, which uses a bias term to threshold the inner product of the attribute vector and the weight vector. The performance of such classifiers depend on the threshold. The receiver operating characteristic (ROC) curve, which is obtained by varying the threshold, is often used as a graphical tool for the analysis of their performance. One of the commonly used summary for the ROC curve is the area under the curve (AUC). There are a number of learning algorithms which can be used to optimize AUC (Cortes and Mohri, 2004; Joachims, 2005). Agarwal et al. (2005) studies the consistency properties for AUC.

We demonstrate that AUC is non-decomposable in the following. AUC is equivalent to the probability that a randomly chosen positive instance has higher score than a randomly chosen negative instance (Hanley, 1982). Thus it measures how good a score function is at distinguishing the positive and negative classes. Let s_i be the score of the i-th instance, and $y_i \in \{0, 1\}$ be the label of the i-th instance, then AUC can be expressed as

$$AUC(\mathbf{s}, \mathbf{y}) = \left(\sum_{y_i=1, y_j=0} \mathrm{I}(s_i > s_j) + \frac{1}{2} \sum_{y_i=1, y_j=0} \mathrm{I}(s_i = s_j) \right) / \sum_{y_i=1} 1 \sum_{y_j=0} 1. \qquad (6.2.7)$$

Proposition 45. *AUC is non-decomposable.*

Proof. Assume AUC is decomposable. Consider the following quantities

$$U_1 = AUC((0,1,3),(0,1,1)) = 1,$$
$$U_2 = AUC((2,1,3),(0,1,1)) = 0.5,$$
$$U_3 = AUC((0,1,3),(0,0,1)) = 1,$$
$$U_4 = AUC((2,1,3),(0,0,1)) = 1.$$

Decomposability implies that $U_1 - U_2 = U_3 - U_4$, but this is false, a contradiction. Hence AUC is non-decomposable. ■

In information retrieval, the perceived value of the retrieved items depend on the relevance of the items and the ranks of the items. NDCG measures the performance of retrieval algorithms for the case when relevance scores are available. Given a collection of n items, let $\mathbf{y} = (y_1, \ldots, y_n)$ be their relevance scores, and let $\mathbf{s} = (s_1, \ldots, s_n)$ be the ranks of them as given by a retrieval algorithm. The discounted cumulative gain (DCG) at rank p is given by

$$DCG_p(\mathbf{s}, \mathbf{y}) = \sum_i \frac{y_i}{d(s_i)} \mathrm{I}(s_i \leq p), \tag{6.2.8}$$

where $d(r)$ is the discount factor for rank r and is often chosen to be $d(1) = 1$, $d(r) = \log(r)$ for $r \geq 2$. Since the value of $DCG_p(\mathbf{s}, \mathbf{y})$ depends on the scale for the relevance scores, the normalized DCG (NDCG) is often used instead. NDCG is obtained by dividing DCG by the maximum possible DCG.

$$NDCG_p(\mathbf{s}, \mathbf{y}) = DCG_p(\mathbf{s}, \mathbf{y}) / \max_{\mathbf{s}'} DCG_p(\mathbf{s}', \mathbf{y}). \tag{6.2.9}$$

Proposition 46. *$NDCG_p$ is non-decomposable.*

Proof. Assume otherwise, consider some $n > p$, then we should have

$$\begin{aligned} & NDCG_p((1,2,\ldots,n),(1,10,\ldots,10^n)) - NDCG_p((1,2,\ldots,n),(2,10,\ldots,10^n)) \\ = \; & NDCG_p((1,2,\ldots,n),(1,\ldots,10^{n-1},20^n)) - NDCG_p((1,2,\ldots,n),(2,10,\ldots,10^{n-1},20^n)), \end{aligned}$$

but this is not the case, a contradiction. ■

6.2.2 Uniform Convergence and Consistency for EUM

Consider an arbitrary classifier $\theta : X \mapsto Y$. Let $F_{\beta,n}(\theta)$ denote the F_β score of θ on D_n. Let $p_{ij,n}(\theta)$ be the empirical probability that a class i instance is observed and predicted as class j by θ; that is,

$$p_{ij,n}(\theta) = \sum_{k=1}^{n} \mathrm{I}(y_k = i \wedge \theta(x_k) = j)/n. \tag{6.2.10}$$

Then we have

$$F_{\beta,n}(\theta) = \frac{(1+\beta^2)p_{11,n}(\theta)}{\beta^2(p_{11,n}(\theta) + p_{10,n}(\theta)) + p_{11,n}(\theta) + p_{01,n}(\theta)}. \tag{6.2.11}$$

Let p_{ij} be the probability that a class i instance is predicted as class j by θ, that is,

$$p_{ij}(\theta) = \mathrm{E}(\mathrm{I}(Y = i \wedge \theta(X) = j)), \tag{6.2.12}$$

Under the i.i.d. assumption, for large i.i.d. sample, the law of large numbers implies that $p_{ij,n}(\theta)$'s converge to $p_{ij}(\theta)$'s. Thus $F_{\beta,n}(\theta)$ is expected to converge to

$$F_\beta(\theta) = \frac{(1+\beta^2)p_{11}(\theta)}{\beta^2\pi_1 + p_{11}(\theta) + p_{01}(\theta)}, \tag{6.2.13}$$

where π_Y denotes $P(Y)$. Hence we can define this to be the F_β-measure of the classifier θ. The above heuristic argument is formalized below. We often omit θ from the notations whenever there is no ambiguity.

Lemma 47. *For any* $\epsilon > 0$, $\lim_{n\to\infty} \mathrm{P}(|F_{\beta,n}(\theta) - F_\beta(\theta)| < \epsilon) = 1$.

Proof. By the law of large numbers, for any $\epsilon_1 > 0$, $\eta > 0$, there exists an N (depending on ϵ_1 and η only) such that for all $n > N$, for any i, j

$$\mathrm{P}(|p_{ij,n} - p_{ij}| < \epsilon_1) > 1 - \eta/3, \tag{6.2.14}$$

Note that only $p_{ij,n}$ is a random variable in the above inequality. Using the union bound, it follows that with probability at least $1 - \eta$, the following hold simultaneously,

$$|p_{11,n} - p_{11}| < \epsilon_1, |p_{10,n} - p_{10}| < \epsilon_1, |p_{01,n} - p_{01}| < \epsilon_1.$$

Let $a = (1+\beta^2)p_{11}$, $b = \beta^2\pi_1 + p_{11} + p_{01}$, $\epsilon_1 = \frac{b\epsilon/(1+\beta^2)}{\frac{2a}{b}+2\epsilon+1}$, then when the above inequalities hold simultaneously, it is easy to verify that $2(1+\beta^2)\epsilon_1 < b$, and

$$\begin{aligned}
\frac{a}{b} - \epsilon &\leq \frac{a - (1+\beta^2)\epsilon_1}{b + 2(1+\beta^2)\epsilon_1} \\
&< \frac{(1+\beta^2)p_{11,n}}{\beta^2(p_{11,n} + p_{10,n}) + p_{10,n} + p_{01,n}}, \\
\frac{a}{b} + \epsilon &\geq \frac{a + (1+\beta^2)\epsilon_1}{b - 2(1+\beta^2)\epsilon_1} \\
&> \frac{(1+\beta^2)p_{11,n}}{\beta^2(p_{11,n} + p_{10,n}) + p_{10,n} + p_{01,n}}.
\end{aligned}$$

That is, $F_\beta(\theta) - \epsilon < F_{\beta,n}(\theta) < F_\beta(\theta) + \epsilon$.

Hence for any $\epsilon > 0$, $\eta > 0$, there exists N such that for all $n > N$, $\mathrm{P}(|F_{\beta,n}(\theta) - F_\beta(\theta)| < \epsilon) > 1 - \eta$. ■

By using a concentration inequality, such as the Hoeffding's inequality, in place of the law of large numbers, we can obtain a bound on the convergence rate.

Lemma 48. *Let* $r(n,\eta) = \sqrt{\frac{1}{2n}\ln\frac{6}{\eta}}$. *When* $r(n,\eta) < \frac{\beta^2\pi_1}{2(1+\beta^2)}$, *then with probability at least* $1-\eta$, $|F_{\beta,n}(\theta) - F_\beta(\theta)| < \frac{3(1+\beta^2)r(n,\eta)}{\beta^2\pi_1 - 2(1+\beta^2)r(n,\eta)}$.

Proof. Let $\eta = 6e^{-2n\epsilon_1^2}$, then $\epsilon_1 = r(n,\eta)$. Using Hoeffding's inequality, for any i,j,

$$\mathrm{P}(|p_{ij,n} - p_{ij}| < \epsilon_1) \quad > 1 - \eta/3. \tag{6.2.15}$$

Let $\epsilon_1 = \frac{\beta^2}{1+\beta^2}\frac{\pi_1\epsilon}{3+2\epsilon}$, then $\epsilon = \frac{3(1+\beta^2)\epsilon_1}{\beta^2\pi_1 - 2(1+\beta^2)\epsilon_1} = \frac{3(1+\beta^2)r(n,\eta)}{\beta^2\pi_1 - 2(1+\beta^2)r(n,\eta)}$. From $\beta^2\pi_1 \le b$ and $\frac{a}{b} \le 1$, it follows that $\epsilon_1 \le \frac{b\epsilon/(1+\beta^2)}{\frac{2a}{b}+2\epsilon+1}$. Similarly as in the proof for Proposition 47, we have $\mathrm{P}(|F_{\beta,n}(\theta) - F_\beta(\theta)| < \epsilon) > 1-\eta$. ■

Lemma 48 leads to the following sample complexity bound.

Corollary 49. *Let* $\epsilon,\eta > 0$, *then when* $n > \frac{1}{2}(\frac{\beta^2}{1+\beta^2}\frac{\pi_1\epsilon}{3+2\epsilon})^{-2}\ln\frac{6}{\eta}$, *with probability at least* $1-\eta$, $|F_{\beta,n}(\theta) - F_\beta(\theta)| < \epsilon$.

The above bounds are not the tightest. For example, Lemma 48 still holds when $\frac{3(1+\beta^2)r(n,\eta)}{\beta^2\pi_1 - 2(1+\beta^2)r(n,\eta)}$ is replaced by the tighter bound $\frac{(1+\beta^2)(2F_\beta(\theta)+1)r(n,\eta)}{\beta^2\pi_1 + p_1(\theta) - 2(1+\beta^2)r(n,\eta)}$, where $p_1(\theta)$ is the probability that θ classifies an instance as positive. In practice, the tighter bound is not useful for estimating the performance of a classifier, because it contains the terms $F_\beta(\theta)$ and $p_1(\theta)$. For the same reason, the tighter bound is also not useful in the uniform convergence that we seek next.

We now show that training to maximize the empirical F_β is consistent, using VC-dimension (Vapnik, 1995) to quantify the complexity of the classifier class.

Theorem 50. *Let* $\Theta \subseteq X \mapsto Y$, $d = VC(\Theta)$, $\theta^* = \arg\max_{\theta\in\Theta} F_\beta(\theta)$, *and* $\theta_n = \arg\max_{\theta\in\Theta} F_{\beta,n}(\theta)$. *Let* $\bar{r}(n,\eta) = \sqrt{\frac{1}{n}(\ln\frac{12}{\eta} + d\ln\frac{2en}{d})}$. *If* n *is such that* $\bar{r}(n,\eta) < \frac{\beta^2\pi_1}{2(1+\beta^2)}$, *then with probability at least* $1-\eta$, $F_\beta(\theta_n) > F_\beta(\theta^*) - \frac{6(1+\beta^2)\bar{r}(n,\eta)}{\beta^2\pi_1 - 2(1+\beta^2)\bar{r}(n,\eta)}$.

Proof. Let $\eta = 12e^{d \ln \frac{2en}{d} - n\epsilon_1^2}$, then $\epsilon_1 = \bar{r}(n, \eta)$. Note that the VC dimension for class consisting of loss functions of the form $\mathrm{I}(y = i \wedge \theta(x) = j)$ is the same as that for Θ, and the same remark applies for the the class consisting of loss functions of the form $\mathrm{I}(\theta(x) = y)$. By (3.3) in (Vapnik, 1995), for any i, j

$$\mathrm{P}(\sup_{\theta} |p_{ij,n}(\theta) - p_{ij}(\theta)| < \epsilon_1) \quad > 1 - \eta/3. \tag{6.2.16}$$

By the union bound, with probability at least $1 - \eta$, the inequalities $\sup_\theta |p_{11,n}(\theta) - p_{11}(\theta)| < \epsilon_1$, $\sup_\theta |p_{10,n}(\theta) - p_{10}(\theta)| < \epsilon_1$, $\sup_\theta |p_{01,n} - p_{01}| < \epsilon_1$, hold simultaneously. Let $\epsilon_1 = \frac{\beta^2}{1+\beta^2} \frac{\pi_1 \epsilon}{3+2\epsilon}$, then following the proof of Lemma 48,

$$\begin{aligned}
& F_\beta(\theta_n) - F_\beta(\theta^*) \\
= \; & F_\beta(\theta_n) - F_{\beta,n}(\theta_n) + F_{\beta,n}(\theta_n) - F_\beta(\theta^*) \\
\geq \; & F_\beta(\theta_n) - F_{\beta,n}(\theta_n) + F_{\beta,n}(\theta^*) - F_\beta(\theta^*) \\
\geq \; & -2\epsilon = -\frac{6(1+\beta^2)\bar{r}(n,\eta)}{\beta^2\pi_1 - 2(1+\beta^2)\bar{r}(n,\eta)}.
\end{aligned}$$

■

The above bound indicates that for smaller π_1 and β, more samples are probably required for convergence to start occurring. When $r(n, \eta) < \frac{\beta^2 \pi_1}{4(1+\beta^2)}$, the difference between $F_{\beta,n}(\theta)$ and $F_\beta(\theta)$ is at most $\frac{6(1+\beta^2)}{\beta^2\pi_1} r(n, \eta)$.

We generalize the technique of obtaining the above consistency proof, and formalize it as a theorem for general loss functions below.

Theorem 51. *Let Θ be a set of functions from X to Y, Suppose $L(\theta)$ is a loss function satisfying the following conditions*

(a) (Well-composed) $L(\theta) = L(L_1(\theta), \ldots, L_k(\theta))$ for some functions $L_1(\theta), \ldots, L_k(\theta)$ such that each L_i satisfies $\mathrm{P}(|L_{i,n}(\theta) - L_i(\theta)| < t(\Theta, n, \eta)) > 1 - \eta$.

(b) (Continuity) $|L(l'_1,\ldots,l'_k)-L(l_1,\ldots,l_k))| < g(\max_i |l'_i-l_i|)$ *for all possible* $(l'_1,\ldots,l'_k)$ *and* $(l_1,\ldots,l_k)$.

If θ^* *is a minimizer of* $L(\theta)$*, then with probability at least* $1-\eta$*,* $L(\theta_n) - L(\theta^*) < g(t(\Theta, n, \frac{\eta}{k}))$.

6.2.3 Optimality of Thresholding in EUM

We now consider a common EUM approach: learning a score function and then using a fixed threshold on the score function. This threshold is obtained by optimizing the F-measure on the training data.

Assume we know the true conditional distribution $P(Y|X)$. Consider the class $\mathcal{T}$ of probability thresholding classifiers of the form $\mathrm{I}_\delta(x) = \mathrm{I}(P(1|x) > \delta)$, and the class $\mathcal{T}'$ containing $\mathrm{I}'_\delta(x) = \mathrm{I}(P(1|x) \geq \delta)$. [2] $\mathcal{T} \cup \mathcal{T}'$ has VC dimension 1, so empirical maximization of F-measure for this class is consistent. Although $\mathcal{T} \cup \mathcal{T}'$ does not contain all possible classifiers on X, an optimal classifier can be found in this class. Let $t^* = \arg\max_{h \in \mathcal{T} \cup \mathcal{T}'} F_\beta(h)$.

Theorem 52. *For any classifier* θ*,* $F_\beta(\theta) \leq F_\beta(t^*)$.

Proof. Let θ be an arbitrary classifier. If $\theta \notin \mathcal{T} \cup \mathcal{T}'$, then when all $x \in X$ are mapped to the number axis using $x \to P(1|x)$, there must be some set B of negative instances which break the positive instances into two sets A and C. Formally, there exist disjoint

[2] Any $\theta \in \mathcal{T}$ can be approximated by members in $\mathcal{T}'$ with arbitrary close F_β, and vice versa, but $\mathcal{T}'$ may contain θ' such that $F_\beta(\theta) \neq F_\beta(\theta')$ for all $\theta \in \mathcal{T}$, implying $\mathcal{T}' \neq \mathcal{T}$. For example, suppose $X = [0,1]$, $P(X = 0.5) = 0.5$, $P(X)$ is uniform on $[0,1] - \{0.5\}$, and $P(1|X) = X$, then $\mathcal{T}$ does not contain a classifier which has the same F_β as $\mathrm{I}'_{0.5}$.

subsets A, B and C of X such that

$$A \cup C = \{x : \theta(x) = 1\},$$

$$\theta(B) = \{0\},$$

$$\sup_{x \in A} P(1|x) \leq \inf_{x \in B} P(1|x) \leq \sup_{x \in B} P(1|x) \leq \inf_{x \in C} P(1|x).$$

Without loss of generality we assume $P(A), P(B), P(C) > 0$. Define

$$\begin{aligned} a &= P(A), \qquad x = \mathrm{E}(P(1|X) \big| X \in A), \\ b &= P(B), \qquad y = \mathrm{E}(P(1|X) \big| X \in B), \\ c &= P(C), \qquad z = \mathrm{E}(P(1|X) \big| X \in C), \end{aligned}$$

then $x \leq y \leq z$. Note that the expectation is taken with respect to X. Let θ_B and θ_C be the same as θ except that $\theta_B(B) = \{1\}$ and $\theta_C(A) = \{0\}$. Thus we have

$$\begin{aligned} F_\beta(\theta) &= \frac{(1+\beta^2)(ax+cz)}{\beta^2\pi_1 + a + c}, \\ F_\beta(\theta_B) &= \frac{(1+\beta^2)(ax+by+cz)}{\beta^2\pi_1 + a + b + c}, \\ F_\beta(\theta_C) &= \frac{(1+\beta^2)cz}{\beta^2\pi_1 + c}. \end{aligned}$$

We show that either $F_\beta(\theta_B) \geq F_\beta(\theta)$ or $F_\beta(\theta_C) \geq F_\beta(\theta)$. Assume otherwise, then

$$F_\beta(\theta) > F_\beta(\theta_B),$$

which implies that

$$ax + cz > (\beta^2\pi_1 + a + c)y.$$

In addition,

$$F_\beta(\theta) > F_\beta(\theta_C),$$

which implies that

$$(\beta^2\pi_1 + c)x > cz.$$

From the above two inequalities, we have

$$ax + cz > (\beta^2\pi_1 + c)x + ax > cz + ax,$$

a contradiction. Hence it follows that we can convert θ to a classifier θ' such that $\theta' \in \mathcal{T} \cup \mathcal{T}'$, and $F_\beta(\theta) \leq F_\beta(\theta') \leq F_\beta(t^*)$. ■

Thresholding is often applied on a score function $f : X \mapsto \mathbf{R}$, rather than on the true conditional distribution. For example, output of a support vector machine is commonly thresholded. Let $f_\delta(x) = \mathrm{I}(f(x) > \delta)$ and $f'_\delta(x) = \mathrm{I}(f(x) \geq \delta)$. Function f is called an *optimal score function* if there is a δ such that $F_\beta(f'_\delta) = F_\beta(t^*)$. We give a sufficient condition for a score function to be optimal. A score function f is *rank-preserving* if it satisfies $f(x_1) > f(x_2)$ iff $P(1|x_1) > P(1|x_2)$ for all $x_1, x_2 \in X$. The sufficient condition relates rank-preservation to optimality:

Theorem 53. *A rank-preserving function is an optimal score function.*

Proof. Immediate from the proof of Theorem 52. ■

By Theorem 53, we can sidestep learning the true distribution and instead try to learn a function which is likely to be rank-preserving. An optimal score function may not be rank-preserving. For example, we can swap the scores of x's above the optimal threshold.

6.2.4 An Asymptotic Equivalence Result

We now investigate the connections between EUM-optimal classifiers and DTA-optimal classifiers when the true distribution $P(X, Y)$ is known. By definition, a DTA-optimal classifier is expected to be better than an EUM-optimal classifier if tested on many i.i.d. test sets. We shall give an asymptotic equivalence result for EUM-optimal classifiers and DTA-optimal classifiers on large i.i.d test sets. In light of Theorem 52, we only need to consider an optimal probability-thresholding classifier as a representative EUM-optimal classifier.

In the following, let $\mathbf{x} = (x_1, \ldots, x_n) \in X^n$ be an i.i.d. sequence of observations. For any classifier θ, let $\theta(\mathbf{x}) = (\theta(x_i))_i$. All expectations, denoted by $\mathrm{E}(\cdot)$, are taken under the conditional distribution $P(\mathbf{y}|\mathbf{x})$. The following theorem says that for an arbitrary classifier θ, when n is large enough, then for any $\mathbf{x}$, the expected F-measure of $\theta(\mathbf{x})$ is close to $F_\beta(\theta)$.

Theorem 54. *For any classifier θ, any $\epsilon, \eta > 0$, there exists $N_{\beta,\epsilon,\eta}$ such that for all $n > N_{\beta,\epsilon,\eta}$, with probability at least $1 - \eta$, $|\mathrm{E}[F_\beta(\theta(\mathbf{x}), \mathbf{y})] - F_\beta(\theta)| < \epsilon$.*

Proof. This follows closely the proof for Lemma 55. ■

In fact, such approximation holds uniformly for the class $\mathcal{T}$.[3]

Lemma 55. *For any $\epsilon, \eta > 0$, there exists $N_{\beta,\epsilon,\eta}$ such that for all $n > N_{\beta,\epsilon,\eta}$, with probability at least $1 - \eta$, for all $\delta \in [0, 1]$, $|\mathrm{E}[F_\beta(\mathrm{I}_\delta(\mathbf{x}), \mathbf{y})] - F_\beta(\mathrm{I}_\delta)| < \epsilon$.*

Proof. $p_i(\delta) = \mathrm{E}(\mathrm{I}(\mathrm{I}_\delta(X) = i))$ denotes the probability that an observation is predicted to be in class i, and $p_{j|i}(\delta) = \mathrm{E}(P(j|X)\big|\mathrm{I}_\delta(x) = i)$ denotes the probability that an observation predicted to be in class i is actually in class j. Let $n_i(\delta) = \sum_k \mathrm{I}(\mathrm{I}_\delta(x_k) = i)$, $n_{ji}(\delta) = \sum_k \mathrm{I}(y_k = j \wedge \mathrm{I}_\delta(x_k) = i)$, then $\tilde{p}_i(\delta) = \frac{n_i}{n}$ and

[3]Both Lemma 55 and Theorem 56 hold for $\mathcal{T} \cup \mathcal{T}'$ as well. We consider $\mathcal{T}$ to simplify the presentation.

$\tilde{p}_{j|i}(\delta) = \frac{n_{ji}(\delta)}{n_i(\delta)}$ are empirical estimates for $p_i(\delta)$ and $p_{j|i}(\delta)$ respectively. We will also need to use $\tilde{p}'_{j|i}(\delta) = \frac{1}{n_i}\sum_i P(j|x)\mathrm{I}(\mathrm{I}_\delta(x) = i)$ as the empirical estimate of $p_{j|i}(\delta)$ based on $\mathbf{x}$ only. Note that $\tilde{p}_i(\delta)$'s and $\tilde{p}'_{j|i}(\delta)$'s are random variables depending on $\mathbf{x}$ only, and $\tilde{p}_{j|i}(\delta)$'s are random variables depending on $\mathbf{x}$ and $\mathbf{y}$. In the following, we shall drop δ from the notations as long as there is no ambiguity. Let $F_\beta(\delta)$ denote the F_β-measure of $\mathrm{I}_\delta(x)$. We have

$$F_\beta(\delta) = \frac{(1+\beta^2)p_1 p_{1|1}}{\beta^2(p_1 p_{1|1} + p_0 p_{1|0}) + p_1}, \tag{6.2.17}$$

$$F_\beta(\mathrm{I}_\delta(\mathbf{x}), \mathbf{y}) = \frac{(1+\beta^2)\tilde{p}_1 \tilde{p}_{1|1}}{\beta^2(\tilde{p}_1 \tilde{p}_{1|1} + \tilde{p}_0 \tilde{p}_{1|0}) + \tilde{p}_1}. \tag{6.2.18}$$

The main idea of the proof is to first show that

(a) there is high probability that $\mathbf{x}$ gives good estimates for $p_i(\delta)$'s and $p_{1|i}(\delta)$'s for all δ, and then show that

(b) for such $\mathbf{x}$, there is high probability that $\mathbf{x}, \mathbf{y}$ give good estimates for $p_i(\delta)$'s and $p_{1|i}(\delta)$'s, thus

(c) $F_\beta(\mathrm{I}_\delta(\mathbf{x}), \mathbf{y})$ has high probability of being close to $F_\beta(\delta)$, and its expectation is close to $F_\beta(\delta)$ as a consequence.

(a) We first show that for any $t > 0$, with probability at least $1 - 12e^{\ln(2en) - nt^4}$, we have for all δ, for all i,

$$|\tilde{p}_i(\delta) - p_i(\delta)| \le t^2, |\tilde{p}_i(\delta)\tilde{p}'_{1|i}(\delta) - p_i(\delta)p_{1|i}(\delta)| \le t^2. \tag{6.2.19}$$

To see this, consider a fixed i. Let $f_\delta(x) = \mathrm{I}(\mathrm{I}_\delta(x) = i)$, $\mathcal{F} = \{f_\delta : 0 \le \delta \le 1\}$, $g_\delta(x) = \mathrm{I}(\mathrm{I}_\delta(x) = i)P(1|x)$, and $\mathcal{G} = \{g_\delta : 0 \le \delta \le 1\}$. Note that the expected value and empirical average of f_δ and g_δ are $p_i(\delta)$, $\tilde{p}_i(\delta)$, $p_i(\delta)p_{1|i}(\delta)$ and $\tilde{p}_i(\delta)\tilde{p}'_{1|i}(\delta)$ respectively. In addition, both $\mathcal{F}$ and $\mathcal{G}$ have VC dimension 1. Thus, by Inequality (3.3) and (3.10) in (Vapnik, 1995), each of the following hold with probability at least

$1 - 4e^{\ln(2en)-nt^4}$,

$$\forall\delta[|\tilde{p}_1(\delta) - p_1(\delta)| \leq t^2]), \tag{6.2.20}$$

$$\forall\delta[|\tilde{p}_i(\delta)\tilde{p}'_{1|i}(\delta) - p_i(\delta)p_{1|i}(\delta)| \leq t^2]. \tag{6.2.21}$$

Now observing that $|\tilde{p}_1(\delta) - p_1(\delta)| \leq t^2$ implies $|\tilde{p}_0(\delta) - p_0(\delta)| \leq t^2$, and applying the union bound, then with probability at least $1 - 12e^{\ln(2en)-nt^4}$, (6.2.19) holds.

(b) Consider a fixed $\mathbf{x}$ satisfying that for some δ, for all i, $|\tilde{p}_i(\delta) - p_i(\delta)| \leq t^2$ and $|\tilde{p}_i(\delta)\tilde{p}'_{1|i}(\delta) - p_i(\delta)p_{1|i}(\delta)| \leq t^2$, we show that if $t < 1$, then with probability at least $1 - 4e^{2nt^3}$,

$$\forall i|\tilde{p}_i(\delta)\tilde{p}_{1|i}(\delta) - p_i(\delta)p_{1|i}(\delta)| \leq 5t. \tag{6.2.22}$$

Consider a fixed i. If $p_i \leq 2t$, then

$$|\tilde{p}_i\tilde{p}_{1|i} - p_ip_{1|i}| \leq \tilde{p}_i\tilde{p}_{1|i} + p_ip_{1|i} \leq \tilde{p}_i + p_i \leq 5t.$$

If $p_i > 2t$, then $|\tilde{p}'_{1|i} - p_{1|i}| \leq t$, [4] and we also have $\tilde{p}_i > 2t - t^2 > t$, that is $n_i > nt$. Note that $\tilde{p}_{1|i}$ is of the form $\frac{1}{n_i}\sum_{i=1}^{n_i} I_i$ where the I_i's are independent binary random variables, and the expected value of $\tilde{p}_{1|i}$ is $\tilde{p}'_{1|x}$, then applying Hoeffding's inequality, with probability at least $1-2e^{-2nt\cdot t^2}$, we have $|\tilde{p}_{1|i}-\tilde{p}'_{1|i}| \leq t$. When $p_i > 2t$, $|\tilde{p}_i - p_i| \leq t^2 < t$, and $|\tilde{p}_{1|i} - \tilde{p}'_{1|i}| \leq t$, we have

$$\begin{aligned}
\tilde{p}_i\tilde{p}_{1|i} - p_ip_{1|i} &\geq (p_i - t)(p_{1|i} - 2t) - p_ip_{1|i} \\
&\geq 2t^2 - 2p_it - p_{1|i}t \geq -5t, \\
\tilde{p}_i\tilde{p}_{1|i} - p_ip_{1|i} &\leq (p_i + t)(p_{1|i} + 2t) - p_ip_{1|i} \\
&\leq 2p_it + p_{1|i}t + 2t^2 \leq 5t.
\end{aligned}$$

[4] This can be seen by observing that if $\tilde{p}'_{1|i} - p_{1|i} > t$, then $\tilde{p}_i\tilde{p}'_{1|i} - p_ip_{1|i} \geq p_i(\tilde{p}'_{1|i} - p_{1|i}) - |\tilde{p}_i - p_i| > 2t \cdot t - t^2 = t^2$, a contradiction. Similarly, the other case can be shown to be impossible.

That is, $|\tilde{p}_i\tilde{p}_{1|i} - p_i p_{1|i}| \leq 5t$. Combining the above argument, we see that (6.2.22) holds with probability at least $1 - 4e^{2nt^3}$.

(c) If for some δ, $\mathbf{x}$ satisfies $|\tilde{p}_i - p_i| \leq t^2 < t$ and $\mathbf{x}, \mathbf{y}$ satisfies (6.2.22), then by Eq. 6.2.18,

$$\begin{aligned} F_\beta(\mathrm{I}_\delta(\mathbf{x}), \mathbf{y}) &\geq \frac{(1+\beta^2)(p_1 p_{1|1} - 5t)}{\beta^2(p_1 p_{1|1} + 5t + p_0 p_{1|0} + 5t) + p_1 + t} \\ &\geq F_\beta(\delta) - \gamma_1 t, \end{aligned}$$

where γ_1 is some positive constant that depends on β and π_1 only. The last inequality can be seen by noting that for $a, b, d, t \geq 0$, $c > 0$, we have $\frac{a-bt}{c+dt} \geq \frac{a}{c} - \frac{ad+bc}{c^2}t$, and observing that in this case $a = (1+\beta^2)p_1 p_{1|1} \leq (1+\beta^2)\pi_1$, $b = 5+5\beta^2$, $c = \beta^2\pi_1 + p_1 \geq \beta^2\pi_1$, and $d = 10\beta^2 + 1$.

Similarly, if $t < \frac{1}{2}\frac{\beta^2\pi_1}{10\beta^2+1}$, then

$$\begin{aligned} F_\beta(\mathrm{I}_\delta(\mathbf{x}), \mathbf{y}) &\leq \frac{(1+\beta^2)(p_1 p_{1|1} + 5t)}{\beta^2(p_1 p_{1|1} - 5t + p_0 p_{1|0} - 5t) + p_1 - t} \\ &\leq F_\beta(\delta) + \gamma_2 t, \end{aligned}$$

where γ_2 is some positive constant that depends on β and π_1 only. The last inequality can be seen by noting that for $a, b, d \geq 0$, $c > 0$, $c > 2dt$, we have $\frac{a+bt}{c-dt} \leq \frac{a}{c} + 2\frac{ad+bc}{c^2}t$, and observing that in this case $a = (1+\beta^2)p_1 p_{1|1} \leq (1+\beta^2)\pi_1$, $b = 5+5\beta^2$, $c = \beta^2\pi_1 + p_1 \geq \beta^2\pi_1$, $d = 10\beta^2 + 1$, and $c > 2dt$.

Now it follows that for an $\mathbf{x}$ satisfying (6.2.19), then for any $\delta \in [0,1]$, for any $t < \frac{1}{2}\frac{\beta^2\pi_1}{10\beta^2+1}$, with probability at least $1 - 4e^{-nt^3}$, $|F_\beta(\mathrm{I}_\delta(\mathbf{x}), \mathbf{y}) - F_\beta(\delta)| \leq \max(\gamma_1, \gamma_2)t$. Hence

$$|\mathrm{E}[F_\beta(\mathrm{I}_\delta(\mathbf{x}), \mathbf{y})] - F_\beta(\delta)| \leq 4e^{-nt^3} \cdot 1 + \max(\gamma_1, \gamma_2)t.$$

For any $\epsilon > 0$, further restrict t to be the maximum satisfying $t \leq \frac{\epsilon}{2\max(\gamma_1,\gamma_2)}$, and let this value be denoted by t_0, then t_0 depends on β, ϵ (and π_1). Now the second term in the above inequality is less than $\epsilon/2$. The first term is monotonically decreasing in n and converges to 0 as $n \to \infty$. Now take $N_{\beta,\epsilon,\eta}$ to be the smallest number such that for $n = N_{\beta,\epsilon,\eta}$, the first term is less than $\epsilon/2$, and $12e^{\ln(2en)-nt^4} < \eta$, then for any $n > N_{\beta,\epsilon,\eta}$, with probability at least $1-\eta$, $|\mathrm{E}_{\mathbf{y}\sim P(\cdot|\mathbf{x})}[F_\beta(\mathrm{I}_\delta(\mathbf{x}),\mathbf{y})] - F_\beta(\delta)| < \epsilon$. ■

The above uniform approximation result leads to the following asymptotic equivalence result.

Theorem 56. *Let* $\mathbf{s}^*(\mathbf{x}) = \max_{\mathbf{s}} \mathrm{E}[F_\beta(\mathbf{s},\mathbf{y})]$, *where maximization is over* $\mathbf{s}$ *satisfying* $\{P(1|x_i) \mid s_i = 1\} \cap \{P(1|x_i) \mid s_i = 0\} = \emptyset$. *Let* $t^* = \arg\max_{t\in\mathcal{T}} F_\beta(t)$. *Then for any* $\epsilon, \eta > 0$,

(a) There exists $N_{\beta,\epsilon,\eta}$ *such that for all* $n > N_{\beta,\epsilon,\eta}$, *with probability at least* $1-\eta$,

$$\mathrm{E}[F_\beta(t^*(\mathbf{x}),\mathbf{y})] \leq \mathrm{E}(F_\beta(\mathbf{s}^*(\mathbf{x}),\mathbf{y})) < \mathrm{E}[F_\beta(t^*(\mathbf{x}),\mathbf{y})] + \epsilon. \tag{6.2.23}$$

(b) There exists $N_{\beta,\epsilon,\eta}$ *such that for all* $n > N_{\beta,\epsilon,\eta}$, *with probability at least* $1-\eta$,

$$|F_\beta(t^*(\mathbf{x}),\mathbf{y}) - F_\beta(\mathbf{s}^*(\mathbf{x}),\mathbf{y}))| < \epsilon. \tag{6.2.24}$$

Proof. (a) By Lemma 55, when $n > N_{\beta,\frac{\epsilon}{2},\eta}$, with probability at least $1-\eta$, $\mathbf{x}$ satisfies that for all δ,

$$|\mathrm{E}_{\mathbf{y}\sim P(\cdot|\mathbf{x})}[F_\beta(\mathrm{I}_\delta(\mathbf{x}),\mathbf{y})] - F_\beta(\delta)| < \epsilon/2.$$

Consider such an $\mathbf{x}$. The lower bound is clear because $\mathbf{s} = \mathrm{I}_{\delta^*}$ satisfies $\{P(1|x_i) : s_i = 1\} \cap \{P(1|x_i) : s_i = 0\} = \emptyset$. For the upper bound, by Theorem 57 and the definition

of $\mathbf{s}^*(\mathbf{x})$, we have $\mathbf{s}^*(\mathbf{x}) = \mathrm{I}_{\delta'}(\mathbf{x})$ for some δ'. Thus

$$\begin{aligned} \mathrm{E}[F_\beta(\mathbf{s}^*(\mathbf{x}), \mathbf{y})] &< F_\beta(\delta') + \epsilon/2 \\ &\leq F_\beta(\delta^*) + \epsilon/2 \\ &< \mathrm{E}[F_\beta(\mathrm{I}_{\delta^*}(\mathbf{x}), \mathbf{y})] + \epsilon. \end{aligned}$$

(b) From the proof for Lemma 55, for any $t > 0$, with probability at least $1 - 12e^{\ln(2en) - nt^4}$, we have for all δ, for all i, $\mathbf{x}$ satisfies (6.2.19), that is,

$$|\tilde{p}_i(\delta) - p_i(\delta)| \leq t^2, |\tilde{p}_i(\delta)\tilde{p}'_{1|i}(\delta) - p_i(\delta)p_{1|i}(\delta)| \leq t^2.$$

In addition, if $t < \frac{1}{2}\frac{\beta^2\pi_1}{10\beta^2+1}$, then for such $\mathbf{x}$, for any δ, with probability at least $1 - 4e^{2nt^3}$,

$$|F_\beta(\mathrm{I}_\delta(\mathbf{x}), \mathbf{y}) - F_\beta(\delta)| < \gamma t,$$

where γ is a constant depending on ϵ (and π_1). Note that there exists δ' such that $\mathrm{I}_{\delta'}(\mathbf{x}) = \mathbf{s}^*(\mathbf{x})$. Using the union bound, with probability at least $1 - 8e^{-2nt^3}$,

$$\begin{aligned} |F_\beta(\mathrm{I}_{\delta'}(\mathbf{x}), \mathbf{y}) - F_\beta(\delta')| &< \gamma t, \\ |F_\beta(\mathrm{I}_{\delta^*}(\mathbf{x}), \mathbf{y}) - F_\beta(\delta^*)| &< \gamma t. \end{aligned} \tag{6.2.25}$$

Hence we have

$$\begin{aligned} \mathrm{E}(F_\beta(\mathrm{I}_{\delta'}(\mathbf{x}), \mathbf{y}) &\leq (1 - 8e^{-2nt^3})(F_\beta(\delta') + \gamma t) + 8e^{-2nt^3}, \\ \mathrm{E}(F_\beta(\mathrm{I}_{\delta^*}(\mathbf{x}), \mathbf{y}) &\geq (1 - 8e^{-2nt^3})(F_\beta(\delta^*) - \gamma t). \end{aligned}$$

Combining the above two inequalities with $\mathrm{E}(F_\beta(\mathrm{I}_{\delta'}(\mathbf{x}), \mathbf{y}) \geq \mathrm{E}(F_\beta(\mathrm{I}_{\delta*}(\mathbf{x}), \mathbf{y})$, we have

$$F_\beta(\delta^*) - F_\beta(\delta') \leq 2\gamma t + \frac{8e^{-2nt^3}}{1 - 8e^{-2nt^3}}.$$

For those $\mathbf{y}$ satisfying (6.2.25), we have

$$\begin{aligned}
&|F_\beta(\mathrm{I}_{\delta'}(\mathbf{x}), \mathbf{y}) - F_\beta(\mathrm{I}_{\delta^*}(\mathbf{x}), \mathbf{y})| \\
&= |F_\beta(\mathrm{I}_{\delta'}(\mathbf{x}, \mathbf{y}) - F_\beta(\delta')| + |F_\beta(\delta') - F_\beta(\delta^*)| \\
&\qquad\qquad + |F_\beta(\delta^*) - F_\beta(\mathrm{I}_{\delta^*}(\mathbf{x}), \mathbf{y})| \\
&< 4\gamma t + \frac{8e^{-2nt^3}}{1 - 8e^{-2nt^3}}.
\end{aligned}$$

Combining the above argument, we have with probability at least $(1-12e^{\ln(2en)-nt^4})(1-8e^{-2nt^3})$ that $|F_\beta(\mathbf{s}^*(\mathbf{x}), \mathbf{y}) - F_\beta(t^*(\mathbf{x}), \mathbf{y})| < 4\gamma t + \frac{8e^{-2nt^3}}{1-8e^{-2nt^3}}$.

Now choose $t = \frac{\epsilon}{8\gamma}$, then for sufficiently large n, we can guarantee that with probability at least $1 - \eta$, $|F_\beta(\mathbf{s}^*(\mathbf{x}), \mathbf{y}) - F_\beta(t^*(\mathbf{x}), \mathbf{y})| < \epsilon$. ■

Part (a) says that the $t^*(\mathbf{x})$ and $\mathbf{s}^*(\mathbf{x})$ have almost the same expected F_β, and Part (b) says that for a large i.i.d. test set $(\mathbf{x}, \mathbf{y})$, t^* and $\mathbf{s}^*$ have almost identical F_β.

The constraint on $\mathbf{s}$ ensures that instances with the same probability of being positive are placed in the same class. In general, optimal predictions may not satisfy this constraint (Lewis, 1995). However, if the underlying distribution satisfies that $P(P(1|X) = \delta) = 0$ for any δ, then the above result is essentially this: given P, an optimal prediction and the prediction using the optimal threshold are asymptotically equivalent. This is demonstrated empirically in Section 6.4.

6.3 Algorithms

We first discuss approximations to EUM, then discuss DTA and present a new efficient prediction algorithm.

6.3.1 Approximations to the EUM Approach

Exact empirical optimization of F-measures for a parametric family is difficult due to its complex piecewise linear nature, and typically only approximations of the F-measures are maximized. We discuss three methods.

In view of the optimality of probability thresholding classifiers, it is natural to first learn an estimate $p(Y|X)$ for $P(Y|X)$, and then learn an optimal threshold δ. If $p(Y|X)$ is chosen from a parametric family using the maximum likelihood (ML) principle, then under very general conditions, the learned distribution follows an asymptotically normal convergence to the model with smallest KL-divergence to the true distribution (White, 1982) (See Section 3.4). Thus when the model family is well-specified, the resulting classifier is asymptotically optimal. We call this the *ML^δ approximation*. Strictly, this is a combination of the conditional probability estimation and F-measure optimization of the threshold, and the convergence rate in Theorem 50 does not apply.

Jansche (2005) learned a logistic regression model $p(Y|X,\phi)$ by maximizing the empirical F_β in Eq. 6.2.1, but with each binary decision s_i replaced by the predictive probabilities $p_i = p(1|x_i,\phi)$. The eventual classifier uses the rule $h(x) = \mathrm{I}(p(1|x,\phi) > 0.5)$. It is unknown whether this method is consistent or whether it follows any asymptotic convergence. There is also no apparent reason to use 0.5 as the threshold, so we shall optimize the threshold on the training data in addition to estimating ϕ. We call this the *F^δ approximation*.

We considered learning a rule $h(x) = \mathrm{I}(p(1|x,\phi) > \delta)$ directly, where ϕ, δ are parameters, by approximating the empirical F_β in Eq. 6.2.1 using $s_i = \mathrm{I}^\gamma(p(1|x_i,\phi) - \delta)$,

where $\mathrm{I}^\gamma(t) = 1/(1+e^{-\gamma t})$ approximates $\mathrm{I}(t > 0)$ for large γ. However, this seemed to overfit easily, and it rarely yielded better performance than the ML^δ and F^δ approximations in our preliminary experiments. We will not consider it further.

6.3.2 Maximizing Expected F-measure

Given a utility function $U(\mathbf{s}, \mathbf{y})$, the decision-theoretic optimal prediction for $\mathbf{x}$ maximizes $\mathrm{E}_{\mathbf{y}\sim P(\cdot|\mathbf{x})}(U(\mathbf{s}, \mathbf{y}))$. In general, the true distribution P is not known and is estimated. The approach that involves first estimating true distributions using maximum likelihood (ML) and then making decision-theoretic optimal predictions will be called the *ML^E approach*. We discuss the two steps in ML^E, then we present an efficient algorithm for computing the optimal predictions.

First, the asymptotic convergence of ML (White, 1982) implies the ML^E approach is asymptotically optimal when estimating with sufficient training examples in a well-specified family. In practice, we will not know whether the model family is well-specified. Nevertheless, as we shall see in Section 6.4.1, the ML^E approach can yield results indistinguishable from the optimal if the model family is misspecified but contains a reasonable approximation to the true distribution.

Second, for arbitrary utility function U, computing the expectation can be computationally difficult. But for the case when the utility function is an F-measure, and $P(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^n P(y_i|x_i)$, efficient algorithms can be designed by exploiting the following characterization of an optimal prediction. Let $p_i = P(1|x_i)$.

Theorem 57. *(Probability Ranking Principle for F-measure, Lewis 1995) Suppose* $\mathbf{s}^* = \max_{\mathbf{s}} \mathrm{E}(F_\beta(\mathbf{s}, \mathbf{y}))$. *Then* $\min\{p_i \mid s_i^* = 1\} \geq \max\{p_i \mid s_i^* = 0\}$.

Thus the decision-theoretic optimal prediction contains the top k instances that are most likely to be positive for some $k \in \{0, \ldots, n\}$. This reduces the number of candidate predictions from 2^n to $n + 1$.

We shall use Lewis's result to first give a new $O(n^3)$ time algorithm for computing the optimal predictions, then improve it to $O(n^2)$ when β^2 is rational, which is often the case. We then illustrate the $O(n^2)$ algorithm for the case when $\beta^2 = 1$.

A Cubic Time Algorithm

Let $F_{\beta;k}(\mathbf{y})$ be the F_β-measure when the first k instances are predicted as positive, then we have

$$F_{\beta;k}(\mathbf{y}) = (1+\beta^2)\sum_{i=1}^{k} y_i / [k + \beta^2 \sum_{i=1}^{n} y_i]. \tag{6.3.1}$$

Let $f_{\beta;k}$ be the expected F_β-measure when the first k instances are predicted as positive, that is

$$f_{\beta;k} = \sum_{\mathbf{y}} P(\mathbf{y}) F_{\beta;k}(\mathbf{y}), \tag{6.3.2}$$

and let $S_{i:j}$ be the number of true positive instances from the i-th to the j-th instances (inclusive), that is

$$S_{i:j} = \sum_{l=i}^{j} y_l. \tag{6.3.3}$$

For $\mathbf{y}$'s satisfying $S_{1:k} = k_1$ and $S_{k+1:n} = k_2$, their F_β's are $(1+\beta^2)k_1/(k+\beta^2(k_1+k_2))$, and the probability this happens is $P(S_{1:k} = k_1)P(S_{k+1:n} = k_2)$, thus

$$f_{\beta;k} = \sum_{\substack{0 \le k_1 \le k \\ 0 \le k_2 \le n-k}} \frac{P(S_{1:k} = k_1)P(S_{k+1:n} = k_2)(1+\beta^2)k_1}{k + \beta^2(k_1+k_2)}. \tag{6.3.4}$$

One can show that $P(S_{1:k} = i)$ and $P(S_{k+1:n} = i)$ are the coefficients of x^i in $\prod_{j=1}^{k}[p_j x + (1-p_j)]$ and $\prod_{j=k+1}^{n}[p_j x + (1-p_j)]$ respectively. Thus, each $f_{\beta;k}$ can be computed in $O(n^2)$ time using $O(n)$ space. Hence computing all $f_{\beta;k}$'s takes $O(n^3)$ time and $O(n)$ space.

A Quadratic Time Algorithm

For rational β^2, we can improve the computation to $O(n^2)$ time and $O(n)$ space. The key is to note that

$$f_{\beta;k} = \sum_{k_1=0}^{k} (1+\beta^{-2}) k_1 P(S_{1:k} = k_1) s(k, k\beta^{-2} + k_1), \tag{6.3.5}$$

where $s(k,\alpha) = \sum_{k_2=0}^{n-k} P(S_{k+1:n} = k_2)/(\alpha + k_2)$. For rational β, the s values required for the $f_{\beta;k}$'s are shared. To compute s, use $s(n,\alpha) = 1/\alpha$, and

$$s(k-1,\alpha) = p_k s(k,\alpha+1) + (1-p_k) s(k,\alpha), \tag{6.3.6}$$

which follows from $P(S_{k:n} = i) = p_k P(S_{k+1:n} = i-1) + (1-p_k) P(S_{k+1:n} = i)$.

To sum up the above discussion, the pseudo-code for the quadratic time algorithm is given in Algorithm 1, with q/r as the reduced fraction of β^2.

Algorithm 1 Compute $f_{\beta;1}, \ldots, f_{\beta;n}$, where $\beta^2 = q/r$

1: For $0 \le i \le n$, set $C[i]$ as the coefficient of x^i in $[p_1 x + (1-p_1)] \ldots [p_n x + (1-p_n)]$;
2: For $1 \le i \le (q+r)n$, $S[i] \leftarrow q/i$;
3: **for** $k = n$ to 1 **do**
4: $\quad f_{\beta;k} \leftarrow \sum_{k_1=0}^{n} (1 + r/q) k_1 C[k_1] S[rk + qk_1]$;
5: $\quad$ Divide C by $p_k x + (1-p_k)$;
6: $\quad$ **for** $i = 1$ to $(q+r)(k-1)$ **do**
7: $\quad\quad S[i] \leftarrow (1-p_k) S[i] + p_k S[i+q]$;

Correctness can be seen by observing that at line 3, $S[i] = s(k, i/q)$, and $C[k_1] = P(S_{1:k} = k_1)$. In practice, polynomial division can be numerically unstable, and it is preferred to precompute all the $C[i]$'s using $O(n^2)$ time and space first.

Illustration for $\beta^2 = 1$

We illustrate the computations done for the quadratic time algorithm for $\beta^2 = 1$.

First we can compute all required $P(S_{1:k} = k_1)$ values as in Figure 6.3.2. We start from the first row, and then each subsequent row can be obtained using values from previous rows only. Each entry is the weighted sum of entries with edges pointing to it. Some weights are labeled on the edges.

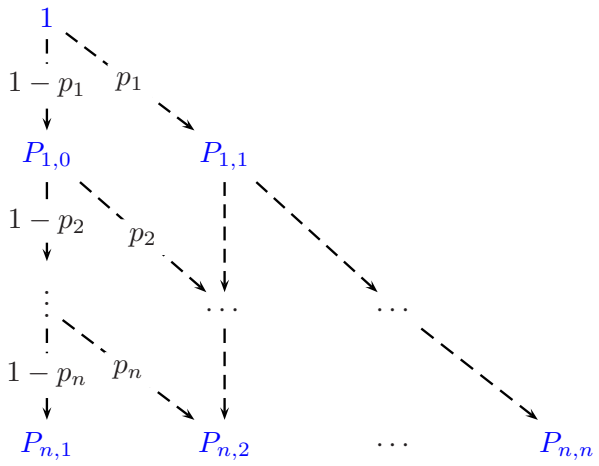


Figure 6.1: Computing all required $P_{k,k_1} = P(S_{1:k} = k_1)$ values.

Then we compute all the required s values as in Figure 6.3.2 with $s(n, i) = 1/i$.

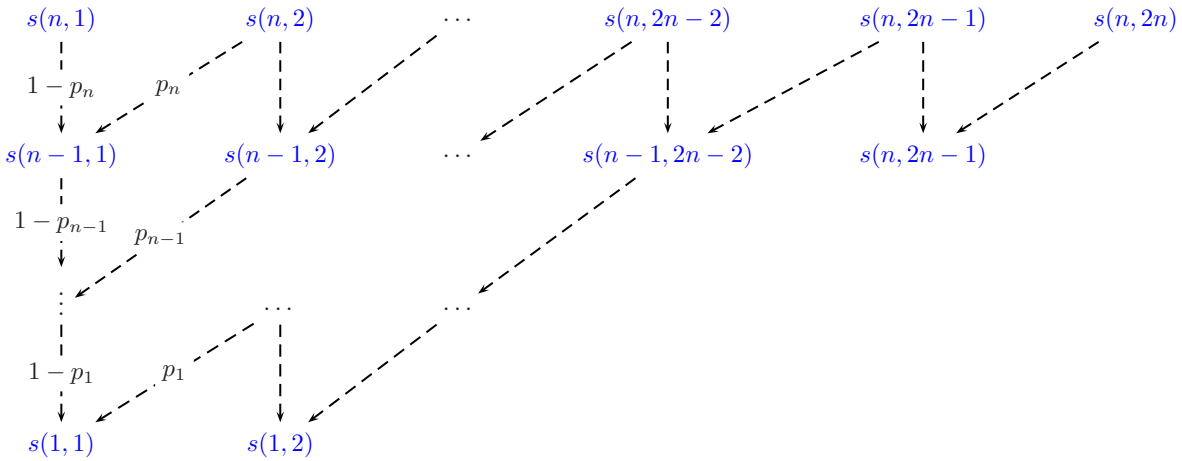


Figure 6.2: Computing all required $s(\cdot, \cdot)$ values.

Now the $f_{\beta;k}$ values can be computed as follows:

$$
\begin{aligned}
f_{\beta;n} &= 0P(S_{1:n}=0)s(n,n) + 2P(S_{1:n}=1)s(n,n+1) + \ldots + 2nP(S_{1:n}=n)s(n,2n), \\
&\ldots \\
f_{\beta;2} &= 0P(S_{1:2}=0)s(2,2) + 2P(S_{1:2}=1)s(2,3) + 4P(S_{1:2}=2)s(2,4), \\
f_{\beta;1} &= 0P(S_{1:1}=0)s(1,1) + 2P(S_{1:1}=1)s(1,2).
\end{aligned}
$$

6.4 Experiments

We empirically demonstrate that EUM can be more robust against model misspecification, but DTA can be better for rare classes on small datasets and a common scenario of domain-adaptation. We use a synthetic dataset, the Reuters-21578 dataset, and four multilabel classification datasets.

6.4.1 Mixtures of Gaussians

We consider a mixture of Gaussians on D dimensions (shown in Figure 6.3): $P(X,Y) = \pi_Y N(X;\mu_Y,\Sigma_Y)$, with $\Sigma_1 = \Sigma_0 = I_D$, $\mu_1 = (S+O)\mathbf{1}/\sqrt{4D}$ and $\mu_0 = -(S-O)\mathbf{1}/\sqrt{4D}$, where S and O are non-negative constants Thus S is the distance between the centers. We shall vary S, O, D, π_1 and the number of training examples N_{tr}. [5] All instances are i.i.d. The optimal F_1 achievable by a classifier θ can be computed (see Eq. 6.2.13), and it depends only on S and π_1. N_{tr} determines how close the estimated distribution is to the optimal model; and the number of test examples, N_{ts}, affects the gap in the performance between the thresholding method and the expectation method (Theorem 56).

[5] These parameters are mainly chosen for simplicity of computation, but the conclusions hold for dataset of similar characteristics as described in the sections below.

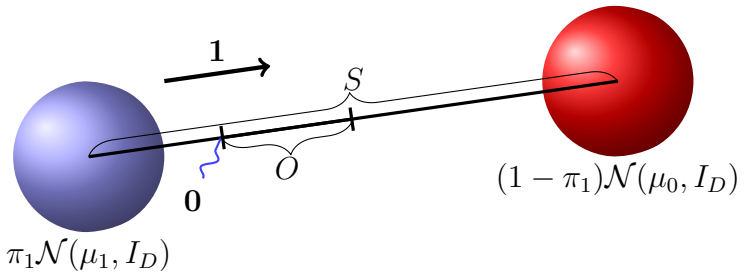


Figure 6.3: Mixture of Gaussians used in the experiments.

We train logistic regression (LR) models using three different attribute vector representations: R_0 consists of the coordinates only, R_1 is R_0 with an additional dummy attribute fixed at 1, and R_2 is R_1 with additional all degree two monomials of the coordinates. LR with R_2 includes the true distribution. The methods compared are ML^δ, F^δ, ML^E, Truth^δ and Truth^E, where last two methods use the true model $P(X, Y)$ for thresholding and expectation.

	ML^E			ML^δ			F^δ					
Setting	R_0	R_1	R_2	R_0	R_1	R_2	R_0	R_1	R_2	Truth^E	Truth^δ	Theory
Default	97.87	97.84	96.02	97.84	97.87	96.15	97.62	97.55	96.37	97.87	97.91	97.72
S=0.4	66.86	66.86	63.77	66.32	66.31	63.55	66.03	66.09	65.72	66.39	65.82	66.88
D=100	94.12	94.14	88.05	94.09	94.08	87.86	95.96	95.98	88.23	97.53	97.53	97.72
N_{tr}=100	95.43	95.48	91.36	94.78	94.69	91.33	95.55	95.34	91.57	97.80	97.36	97.72
π_1=0.05	75.19	90.79	84.07	91.84	90.17	84.21	92.36	89.56	85.21	92.72	92.26	91.73
O=50	66.01	67.83	96.10	65.44	89.29	96.10	97.04	96.88	97.41	97.87	97.91	97.72

Table 6.1: Performance of different methods for optimizing F_1 on mixtures of Gaussians

The first column in Table 6.1 lists the parameter settings. For the row headed by *Default*, we use $D = 10$, $S = 4$, $O = 0$, $N_{tr} = 1000$, $N_{ts} = 3000$, and $\pi_1 = 0.5$. This

dataset is low dimensional, almost noiseless, balanced and has sufficiently many train and test instances.[6] Each of the remaining rows uses the same set of parameters, except for the one parameter indicated on the first column. LR with R_0 or R_1 contains a good approximation to the true distribution for all settings except $\pi_1 = 0.05$ and $O = 50$. For $\pi_1 = 0.05$, the class is imbalanced and such imbalance cannot be modelled without the dummy attribute. Thus R_0 will not give a good model, but R_1 will. For $O = 50$, the centers are far from the origin, and this makes both R_0 and R_1 inadequate for density estimation.

In Table 6.1, the F_1 results for Truth^E, Truth$^\delta$ and Theory (the theoretical optimal F_1 for a classifier, computed using Eq. 6.2.13) are similar. These are expected according to Theorem 56. Most other scores are close to the optimal scores. For ML^E and ML$^\delta$, these scores are expected due to the presence of a good approximation to the true distribution in the model family, and the asymptotic convergence property of ML$^\delta$ and ML^E given sufficiently many examples, as discussed in Section 6.3. For F$^\delta$, although we lack its theoretical convergence to an optimal classifier, the results suggest that such convergence may hold.

The scores obtained using R_2 are generally lower than scores obtained using R_0 and R_1 under the settings *Default*, $S = 0.4$, $D = 100$, and $N_{tr} = 100$, though R_2 gives a well-specified model class while R_1 and R_0 do not. Thus, a well-specified model class is not necessarily better. This is because a misspecified model class with a small VC dimension can converge to the optimal model within the class using fewer samples than a well-specified model class with a higher VC dimension. To choose a class of the right complexity, one may follow the structural risk minimization principle (Vapnik, 1995). This requires bounds like those in Lemma 48 and Theorem 50. However, the given bounds cannot be used because they only apply for large samples.

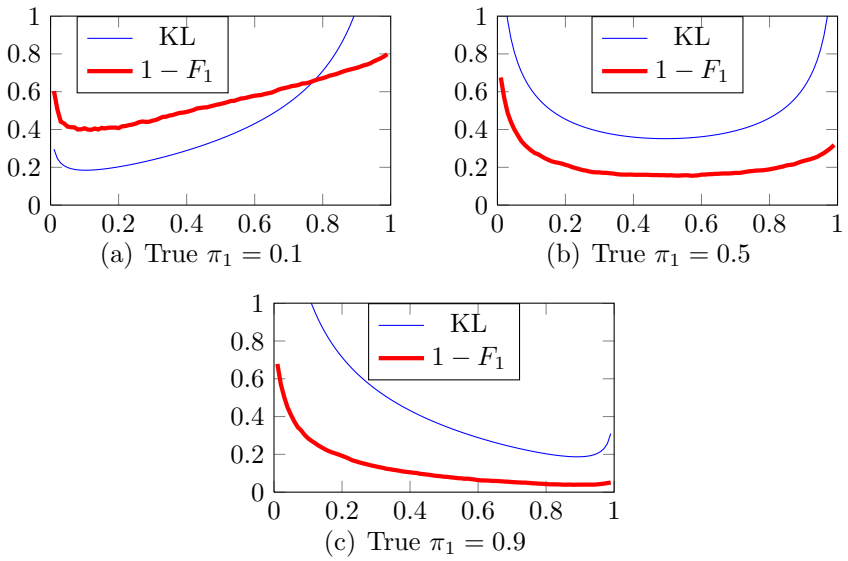
The gaps between R_2 scores and the optimal score for *Default* is significantly smaller

[6] We have verified that the sizes are large enough to give the same conclusions for other i.i.d. data of the same sizes.

than the gaps for $S = 0.4$, $D = 100$, $N_{tr} = 100$, and $\pi_1 = 0.05$. This suggests that higher noise level, higher model class complexity, smaller training size, and smaller positive ratio make it harder to learn a good classifier. Note that Theorem 50 already suggests that in EUM, smaller positive ratio can make learning more difficult.

For the setting $\pi_1 = 0.05$, using R_0, ML^E performs poorly, while ML^δ is close to optimal. ML^E's poor performance is expected due to poor quality of the learned distribution, and ML^δ's performance can be justified by Theorem 53: the thresholding method can remain optimal when the score function is rank-preserving but not close to the true probability distribution. For the setting $O = 50$, both ML^E and ML^δ perform poorly using R_0, but ML^δ is much better than ML^E using R_1. Thus although ML^δ can still be severely affected by model misspecification, it is still relatively robust. In addition, for $\pi_1 = 0.05$ and $O = 50$, F^δ has much higher or at least comparable scores than ML^E and ML^δ. This suggests that *if the model class is severely misspecified, then EUM can be more robust than DTA.*

We also compare ML^E and ML^δ on small test sets with $N_{ts} = 100$ (Theorem 56 only holds for large test set size). We observed similar performances from ML^δ and ML^E when π_1 is high, but ML^E seems significantly better than ML^δ when π_1 is small. To illustrate, Table 6.2 gives the results when the same setting as $\pi_1 = 0.05$ in Table 6.1 is used to generate the data. It shows that, *with a sufficiently accurate model, ML^E can be better than ML^δ and F^δ on rare classes.*



(a) True $\pi_1 = 0.1$ (b) True $\pi_1 = 0.5$

(c) True $\pi_1 = 0.9$

Figure 6.4: Effect of the quality of probability model on the decision-theoretic method. The x-axes are the π_1 values for the assumed distribution, and the y-axes are the corresponding $1 - F_1$ and KL values.

	ML^E			ML^δ			F^δ				
	R_0	R_1	R_2	R_0	R_1	R_2	R_0	R_1	R_2	Truth^E	Truth^δ
Mean	36.70	63.00	58.80	60.78	62.01	58.31	61.40	59.98	53.16	63.32	60.71
Std. dev.	13.04	20.67	21.49	23.69	21.34	21.87	22.18	22.04	23.02	20.46	23.72

Table 6.2: The means and standard deviations of the F_1 scores in percentage, computed using 2000 i.i.d. trials, each with test set of size 100, for mixtures of Gaussians with $D = 10$, $S = 4$, $O = 0$, $N_{tr} = 1000$ and $\pi_1 = 0.05$.

Effect of Model Quality

We also perform experiments to study the effect of incorrect probability models on ML^E. We use the *Default* setting in the previous section, with $\pi_1 = 0.5$ and $S = 4$ changed to $S = 2$, as the true distribution, to generate a set of 3000 i.i.d. test instances. We make optimal predictions using an assumed distribution which is the same as the true one except that we vary π_1. For each π_1, we compute the F_1 and

the Kullback-Leibler-divergence (KL) from the true to the assumed distribution on the test set. These are plotted in Figure 6.4(b), where $1 - F_1$ is plotted instead of F_1. Figures 6.4(a) and 6.4(c) plot for similar experiments, but using 0.1 and 0.9 as the true π_1 instead. Our choice of $S = 2$ instead of $S = 4$ for the true distribution has made the difference between the true and assumed distributions more pronounced in the plots. Comparing the curves for KL and $1 - F_1$ within each figure, we see that the F-measure of DTA is roughly positively correlated with the model quality. The plot for $1 - F_1$ in Figure 6.4(a) exhibits higher curvature around the true π_1 than those in the other two figures. This suggests that if the true distribution has a small positive ratio, the performance is more sensitive to model quality.

Domain Adaptation

In domain adaptation, the test distribution differs from the training one. One common scenario is when $P(X)$ changes but $P(Y|X)$ does not. Using the mixture of Gaussians with $D = 10$, $S = 4$, $O = 0$ and $\pi_1 = 0.5$, we generate 5000 i.i.d. training instances, and 5000 test instances with $P(Y|X) < 0.5$. The F_1 scores for Truth$^\delta$, Truth^E, ML$^\delta$ and ML^E (using R_1) are 21%, 38%, 11% and 36% respectively. Similar results are obtained under similar settings. Under such conditions, DTA is more robust than EUM.

6.4.2 Text Classification

We evaluate on the Reuters-21578 dataset[7] using the ModApte partition, which has 9603 training documents and 3299 test documents. We train two models: the standard multinomial naïve Bayes (NB) model and a LR model, using word occurrence counts and a dummy attribute fixed at one. Both models are regularized. For NB, we use the Laplace corrector with one count for class and word counts. For LR, we use the Gaussian norm on the parameters. We use only those topics with at least C positive

[7] Available at http://www.daviddlewis.com/resources/testcollections/reuters21578/.

instances in both the train and test sets, and we vary C. Table 6.3 reports macro-F_1 scores (the F_1 averaged over topics), where $\text{ML}^{.5}$ uses 0.5 to threshold the probabilities,

	Naïve Bayes			Logistic regression			
C	$\text{ML}^{.5}$	ML^{δ}	ML^{E}	$\text{ML}^{.5}$	ML^{δ}	F^{δ}	ML^{E}
1	17.4	17.7	17.7	35.8	36.6	37.3	39.9
10	29.8	30.1	30.2	55.1	56.4	57.2	57.6
50	69.9	69.1	70.1	75.2	75.7	76.6	75.6
100	73.7	73.5	73.7	75.5	75.9	76.5	75.8

Table 6.3: Macro-F_1 scores in percentage on the Reuters-21578 dataset, computed for those topics with at least C positive instances in both the training and test sets. The number of topics down the rows are 90, 50, 10 and 7.

In Table 6.3, although NB generally does not provide good probability estimates, ML^{E} is still at least comparable to $\text{ML}^{.5}$ and ML^{δ}. With LR, ML^{E} is a few percents better for rare classes. Chai (2005) used Gaussian process and obtained similar conclusion.

6.4.3 Multilabel Datasets

We evaluate on four standard multilabel classification datasets.[8] We train regularized LR, with the regularization parameter for each class selected using two fold cross validation. Macro-F_1 scores are shown in Table 6.4. The bracketed scores are obtained by choosing the regularization parameter giving a model with minimum empirical KL divergence on the test data. Each bracketed score is higher than its non-bracketed counterpart, thus models closer to the true one perform better for both ML^{E} and ML^{δ}.

[8]These are available at http://mulan.sourceforge.net/.

Comparing the scores for ML^E with those for ML^δ and F^δ, bracketed or not, we see that ML^E performs better, especially for smaller C, suggesting ML^E is better for rare classes.

C	T	ML^δ	ML^E	F^δ
yeast (1500 train, 917 test)				
1	14	47.14 (47.54)	48.16 (48.47)	46.61
50	13	50.76 (50.34)	51.38 (51.71)	50.19
300	5	73.79 (73.31)	73.52 (73.74)	73.71
medical (645 train, 333 test)				
1	32	48.88 (51.93)	51.48 (53.91)	48.45
10	12	84.81 (84.49)	85.19 (85.84)	86.01
50	2	87.62 (88.78)	90.12 (88.99)	91.56
scene (1211 train, 1196 test)				
1	6	68.80 (70.50)	68.57 (70.80)	69.05
100	6	68.80 (70.50)	68.57 (70.80)	69.05
enron (1123 train, 579 test)				
1	52	19.70 (25.53)	21.61 (26.45)	19.24
10	26	35.26 (38.00)	38.76 (39.74)	35.86
50	9	59.21 (59.82)	60.15 (60.44)	61.60

Table 6.4: Macro-F_1 scores in percentage on four multilabel datasets, computed for those T labels with at least C positive instances in both the training and test sets.

6.5 Discussion

We gave theoretical justifications and connections for optimizing F-measures using EUM and DTA. We empirically demonstrated that EUM seems more robust against model misspecification, while given a good model, DTA seems better for handling rare classes and a common domain adaptation scenario.

A few important questions are unanswered yet: existence of interesting classifiers for which EUM can be done exactly, quantifying the effect of inaccurate models on optimal predictions, identifying conditions under which one method is preferable to another, and practical methods for selecting the best method on a dataset. Results presented here only hold for large data sets, and it is important to consider the case for small number of instances. Experiments with and analyses of other methods may yield additional insights as well.

Chapter 7

Conclusion

This thesis is motivated by the search for a more general framework towards statistical learning in the context of increasing need in dealing with structure in data and increasing popularity in performance measures other than those well-understood decomposable losses such as accuracy and square loss. Its main contributions consist of exact polynomial time inference and learning algorithms for a class of sparse high-order CRF and a class of sparse FCRFs, and the theory and algorithms for optimizing F measures, a class of popular non-decomposable performance measures. There are various directions along which the works can be further developed to move towards a general framework for handling structures and non-decomposable losses. We discuss a few below.

First, both types of sparse CRFs are special cases of CRFs with sparse potential functions, and they have their own limitations as pointed out in the discussions at the end of Chapter 4 and Chapter 5. Designing efficient algorithms for general CRFs with sparse potential functions will give us a tool to handle a rich class of structural dependencies.

Second, it will be interesting to have efficient methods to detect or generate sparse structures (as compared to ℓ-1-type algorithms). For example, for our high-order CRFs,

if the number of patterns is large, how to select only the most important ones so as to get a good sparse approximation? Our FCRFs allow the exploitation of sparsity of patterns of the outputs of reasonable baselines, and can be viewed as an automatic way of converting a problem with dense structures to one with sparse structures. However this may not work in general, for example, using baseline outputs is not likely to work very well for high-order CRFs, because our high-order CRFs use all patterns occurring at all time steps, so we may still have many patterns.

Third, is it possible to exploit algorithms for sparse models to better handle dense models? This is actually related to the question above. A simple approach is to first reduce a dense model to a sparse one, such as ignoring potential functions with small weight, so as to have a sparse model. After that, inference can be done with respect to the sparse model, and the results can then be used in learning. This is something that can be carried out as a preliminary study.

Fourth, our investigation on non-decomposable performance measures focus on F-measures. Connections to results on other types of non-decomposable performance measures, such as AUC, should be examined to see whether more general theory and algorithms are possible.

Last, this thesis has not explored how we can deal with both structures in data and non-decomposable losses at the same time. Attempts to try to efficiently compute predictions with maximal F-measure for a collection of sequences using CRFs as the underlying model turn out to be much more challenging. Results along this line will be very interesting.

Index

Bibliography

A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59, 1994.

S. Agarwal, T. Graepel, R. Herbrich, S. Har-Peled, and D. Roth. Generalization bounds for the area under an ROC curve. *JMLR*, 6:393–425, 2005.

John Aldrich. R.A. Fisher and the making of maximum likelihood 1912-1922. *Statistical Science*, 12(3):162–176, 1997.

C. Andrieu, N. De Freitas, A. Doucet, and M.I. Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1):5–43, 2003.

M. Anthony and P.L. Bartlett. *Neural network learning: Theoretical foundations*. Cambridge University Press, 1999.

A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22:39–71, 1996.

R.H. Berk. Limiting behavior of posterior distributions when the model is incorrect. *The Annals of Mathematical Statistics*, 37:51–58, 1966.

R.H. Berk. Consistency a posteriori. *The Annals of Mathematical Statistics*, 41:894–906, 1970.

J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195, 1975.

A.L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97:245–271, 1997.

L. Bottou. Stochastic learning. *Advanced lectures on machine learning*, pages 146–168, 2004.

S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge Univ Pr, 2004.

A.P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Eric Brill. Some advances in transformation-based part of speech tagging. In *AAAI*, pages 722–727, 1994.

A.M. Bronstein, M.M. Bronstein, and R. Kimmel. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Sciences of the United States of America*, 103(5):1168–1172, 2006.

Kian Ming Adam Chai. Expectation of F-measures: tractable exact computation and some empirical observations of its properties. In *SIGIR*, pages 593–594, 2005.

C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. *NIPS*, 16(16): 313–320, 2004.

A. Culotta, D. Kulp, and A. McCallum. Gene prediction with conditional random fields. Technical Report UM-CS-2005-028, University of Massachusetts, Amherst, 2005.

G. Darmois. Sur les lois de probabilité á estimation exhaustive. *C.R. Acad. Sci*, 200: 1265–1266, 1935.

J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43:1470–1480, 1972.

H. Daumé, J. Langford, and D. Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19:380–393, 2002.

K. Dembczynski, W. Waegeman, W. Cheng, and E. Hullermeier. An exact algorithm for F-measure maximization. In *NIPS*, 2011.

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. Series B (Methodological)*, 39(1):1–38, 1977.

L. Devroye. Any discrimination rule can have an arbitrarily bad probability of error for finite sample size. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2:154–157, 1982.

T. G. Dietterich, Adam Ashenfelter, and Y. Bulatov. Training conditional random fields via gradient tree boosting. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

Thomas G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

Kevin Duh. Jointly labeling multiple sequences: a factorial HMM approach. In *Proceedings of the ACL Student Research Workshop*, ACLstudent '05, pages 19–24, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

T. Ellman. Explanation-based learning: A survey of programs and perspectives. *ACM Computing Surveys (CSUR)*, 21(2):164–221, 1989.

R.E. Fan and C.J. Lin. A study on threshold selection for multi-label classification. Technical report, Department of Computer Science, National Taiwan University, 2007.

T. Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

V. Feldman, V. Guruswami, P. Raghavendra, and Y. Wu. Agnostic learning of monomials by halfspaces is hard. In *FOCS'09*, pages 385–394. IEEE, 2009.

Ronald A. Fisher. On an absolute criterion for fitting frequency curves. *Messenger of Mathmatics*, 41:155–160, 1912.

Ronald A. Fisher. On the "probable error" of a coefficient of correlation deduced from a small sample. *Metron*, 1:3–32, 1921.

Ronald A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222(594-604):309–368, 1922.

Ronald A. Fisher. Two new properties of mathematical likelihood. *Proc. Roy. Soc., A*, 144:285–307, 1934.

Y. Freund and R.E. 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.

Z. Ghahramani. Unsupervised learning. *Advanced Lectures on Machine Learning*, pages 72–112, 2004.

Zoubin Ghahramani and Michael I. Jordan. Factorial Hidden Markov Models. *Machine Learning*, 29(2-3):245–273, 1997.

E.M. Gold. Language identification in the limit. *Information and control*, 10(5):447–474, 1967.

D. E. Goldberg. Genetic and evolutionary algorithms come of age. *Communications of the ACM*, 37(3):113–119, 1994.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

J.M. Hammersley and P. Clifford. Markov Fields on Finite Graphs and Lattices, 1971. Unpublished manuscript.

J.A. Hanley. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 743:29–36, 1982.

T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2005.

S. Haykin. *Neural networks: a comprehensive foundation*. Prentice hall, 1999.

C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.

P.J. Huber. The behavior of maximum likelihood estimates under nonstandard conditions. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1(1):221–233, 1967.

Sorin Istrail. Statistical mechanics, three dimensionality and NP-completeness I: Universality of intractability for the partition function of the Ising model across non-planar lattices. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 87–96, 2000.

M. Jansche. Maximum expected F-measure training of logistic regression models. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 692–699, 2005.

M. Jansche. A maximum expected utility framework for binary sequence labeling. In *ACL*, 2007.

K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

E. T. Jaynes. Information theory and statistical mechanics i. *Phys. Rev*, 106:620–630, 1957a.

E. T. Jaynes. Information theory and statistical mechanics ii. *Phys. Rev*, 108:171–190, 1957b.

Frederick Jelinek, John D. Lafferty, and Robert L. Mercer. Basic methods of probabilistic context free grammars. *Speech Recognition and Understanding. Recent Advances, Trends, and Applications*, page Springer Verlag, 1992.

F. Jiao, S. Wang, C.H. Lee, R. Greiner, and D. Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 209–216. Association for Computational Linguistics, 2006.

T. Joachims. A support vector method for multivariate performance measures. In *ICML*, pages 377–384, 2005.

Mark Johnson. The DOP estimation method is biased and inconsistent. *Computational Linguistics*, 28(1):71–76, 2002.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37 (2):183–233, 1999.

R. H. Kassel. *A comparison of approaches to on-line handwritten character recognition.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1995.

S. Kim, Y. Song, K. Kim, J.W. Cha, and G.G. Lee. MMR-based active machine learning for bio named entity recognition. In *Proceedings of the Human Language*

Technology Conference of the NAACL, Companion Volume: Short Papers, pages 69–72. Association for Computational Linguistics, 2006.

R. Kindermann and J.L. Snell. Markov random fields and their applications. *American Mathematical Society Providence, Rhode Island*, 1980.

T. Koo, A. Globerson, X. Carreras, and M. Collins. Structured prediction models via the matrix-tree theorem. In *Proc. EMNLP*, 2007.

B. O. Koopman. On distributions admitting a sufficient statistic. *Transactions of the American Mathematical Society*, 39:399–409, 1936.

SB Kotsiantis, ID Zaharakis, and PE Pintelas. Supervised machine learning: A review of classification techniques. *Frontiers in Artificial Intelligence and Applications*, 160: 3, 2007.

F.R. Kschischang, B.J. Frey, and H.A. Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519, 2001.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

P. Langley and H.A Simon. Applications of machine learning and rule induction. *Communications of the ACM*, 38(11):54–64, 1995.

L.M. Le Cam. On some asymptotic properties of maximum likelihood estimates and related bayes' estimates. *University of California Publication in Statistics*, 1:277–330, 1953.

D.D. Lewis. Evaluating and Optimizing Autonomous Text Classification Systems. In *SIGIR*, pages 246–254, 1995.

Y. Li, Y. Tian, L.Y. Duan, J. Yang, T. Huang, and W. Gao. Sequence multi-labeling: a unified video annotation scheme with spatial and temporal context. *Multimedia, IEEE Transactions on*, 12(8):814–828, 2010.

D.V. Lindley. *Bayesian statistics*. Society for Industrial and Applied Mathematics, 1972.

B.G. Lindsay. Composite likelihood methods. *Contemporary Mathematics*, 80(1):221–239, 1988.

D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Gideon S. Mann and Andrew McCallum. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *HLT-NAACL (Short Papers)*, pages 109–112, 2007.

C.D. Manning, P. Raghavan, and H. Schutze. *Introduction to information retrieval*. Cambridge University Press Cambridge, 2009.

A. McCallum. Efficiently inducing features of conditional random fields. In *Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI03)*, pages 403–410, 2003.

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 591–598, 2000.

Kevin P. Murphy and Mark A. Paskin. Linear-time inference in hierarchical HMMs. In *Advances in Neural Information Processing Systems 14*, 2002.

Kevin P Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999.

Viet Cuong Nguyen, Nan Ye, Wee Sun Lee, and Hai Leong Chieu. Semi-Markov conditional random field with high-order features. In *ICML Workshop on Structured Sparsity: Learning and Inference*, 2011.

D.J. Patterson, D. Fox, H. Kautz, and M. Philipose. Fine-grained activity recognition by aggregating abstract object usage. In *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*, pages 44–51. IEEE, 2005.

Judea Pearl. Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach. In *AAAI*, pages 133–136, 1982.

K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, 2(11):559–572, 1901.

J. Petterson and T. Caetano. Reverse multi-label learning. In *NIPS*, pages 1912–1920, 2010.

D. Pinto, A. McCallum, X. Wei, and W.B. Croft. Table extraction using conditional random fields. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 235–242, 2003.

E. J. Pitman. Sufficient statistics and intrinsic accuracy. *Proceedings of the Cambridge Philosophical Society*, 32:1936, 567-579.

Xian Qian, Xiaoqian Jiang, Qi Zhang, Xuanjing Huang, and Lide Wu. Sparse higher order conditional random fields for improved sequence labeling. In *ICML*, page 107, 2009.

A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *NIPS*, 2004.

J. Ross Quinlan. *C4. 5: programs for machine learning*. Morgan Kaufmann, 1993.

Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

S. Robertson. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, 60(5):503–520, 2004.

F. Rosenblatt. *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. Spartan Books, Washington D.C, 1962.

Carsten Rother, Vladimir Kolmogorov, Victor Lempitsky, and Martin Szummer. Optimizing binary MRFs via extended roof duality. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels: : Support vector machines, regularization, optimization, and beyond*. MIT Press, 2002.

Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *COLT/EuroCOLT*, pages 416–426, 2001.

Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *NAACL'03*, pages 134–141, 2003.

Yoram Singer Shai Fine and Naftali Tishby. The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, pages 32(1):41–62, 1998.

C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(379-423):623–656, July, October 1948.

Ray J. Solomonoff. A formal theory of inductive inference. Part I. *Information and control*, 7(1):1–22, 1964.

W.M. Soon, H.T. Ng, and D.C.Y. Lim. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 27(4):521–544, 2001.

K. Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

C.J. Stone. Consistent nonparametric regression. *The annals of statistics*, 5(4):595–620, 1977.

Charles Sutton. GRMM: GRaphical Models in Mallet, 2006. http://mallet.cs.umass.edu/grmm/.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *The Journal of Machine Learning Research*, 8:693–723, 2007.

J. Suzuki, E. McDermott, and H. Isozaki. Training conditional random fields with multivariate evaluation measures. In *ACL*, pages 217–224, 2006.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In *NIPS*, 2003.

J.B. Tenenbaum, V. De Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

Andrey Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl*, 5:1035–1038, 1963.

E.F. Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *HLT-NAACL*, pages 142–147, 2003.

T.T. Truyen, D.Q. Phung, H.H. Bui, and S. Venkatesh. Hierarchical semi-Markov conditional random fields for recursive sequential data. In *NIPS*, 2008.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, pages 104–112, 2004.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6(2):1453–1484, 2005.

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11): 1134–1142, 1984.

C.J. van Rijsbergen. Foundation of evaluation. *Journal of Documentation*, 30(4): 365–373, 1974.

V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 1995.

Vladimir N. Vapnik. *Statistical learning theory*. Wiley-Interscience, 1998.

Vladimir N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

Martin J. Wainwright, Tommi Jaakkola, and Alan S. Willsky. Tree-based reparameterization for approximate inference on loopy graphs. In *NIPS*, pages 1001–1008, 2001.

M.J. Wainwright. Estimating the wrong graphical model: Benefits in the computation-limited setting. *The Journal of Machine Learning Research*, 7:1829–1859, 2006.

M.J. Wainwright and M.I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1:1–305, 2008.

A. Wald. Note on the consistency of the maximum likelihood estimate. *The Annals of Mathematical Statistics*, 20(4):595–601, 1949.

P. Werbos. *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Department of Applied Mathematics, Harvard University, Cambridge Mass., 1974.

H. White. Maximum likelihood estimation of misspecified models. *Econometrica: Journal of the Econometric Society*, pages 1–25, 1982.

D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996a.

D. H. Wolpert. The existence of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1391–1420, 1996b.

Y. Yang. A study of thresholding strategies for text categorization. In *SIGIR*, pages 137–145, 2001.

Nan Ye, Wee Sun Lee, Hai Leong Chieu, and Dan Wu. Conditional random fields with high-order features for sequence labeling. In *NIPS*, 2009.

Nan Ye, Kian Ming Adam Chai, Wee Sun Lee, and Hai Leong Chieu. Optimizing f-measures: A tale of two approaches. In *ICML*, 2012.

J.S. Yedidia, W.T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, 8:236–269, 2003.

D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3:1083–1106, 2003.

N.L. Zhang and D. Poole. A simple approach to Bayesian network computations. In *Proceedings of the biennial conference-Canadian society for computational studies of intelligence*, pages 171–178, 1994.

X. Zhang, T. Graepel, and R. Herbrich. Bayesian online learning for multi-label and multi-variate performance measures. In *AISTATS*, pages 956–963, 2010.

Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.