LEARNING STRUCTURED PREDICTION MODELS: A LARGE MARGIN APPROACH

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Ben Taskar
December 2004

© Copyright by Ben Taskar 2005
All Rights Reserved

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Daphne Koller
Computer Science Department
Stanford University
(Principal Advisor)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Andrew Y. Ng
Computer Science Department
Stanford University

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Fernando Pereira
Computer Science Department
University of Pennsylvania

Approved for the University Committee on Graduate Studies:

To my parents, Mark and Tsilya, and my love, Anat.

Abstract

Most questions require more than just true-false or multiple-choice answers. Yet supervised learning, like standardized testing, has placed the heaviest emphasis on complex questions with simple answers. The acquired expertise must now be used to address tasks that demand answers as complex as the questions. Such complex answers may consist of multiple interrelated decisions that must be weighed against each other to arrive at a globally satisfactory and consistent solution to the question. In natural language processing, we often need to construct a global, coherent analysis of a sentence, such as its corresponding part-of-speech sequence, parse tree, or translation into another language. In computational biology, we analyze genetic sequences to predict 3D structure of proteins, find global alignment of related DNA strings, and recognize functional portions of a genome. In computer vision, we segment complex objects in cluttered scenes, reconstruct 3D shapes from stereo and video, and track motion of articulated bodies.

We typically handle the exponential explosion of possible answers by building models that compactly capture the structural properties of the problem: sequential, grammatical, chemical, temporal, spatial constraints and correlations. Such structured models include graphical models such as Markov networks (Markov random fields), recursive language models such as context free grammars, combinatorial optimization problems such as weighted matchings and graph-cuts. This thesis presents a discriminative estimation framework for structured models based on the large margin principle underlying support vector machines. Intuitively, the large-margin criterion provides an alternative to probabilistic, likelihood-based estimation methods by concentrating directly on the robustness of the decision boundary of a model. Our framework defines a suite of efficient learning algorithms that rely on the expressive power of convex optimization to compactly capture inference or

solution optimality in structured models. For some of these models, alternative estimation methods are intractable.

The largest portion of the thesis is devoted to Markov networks, which are undirected probabilistic graphical models widely used to efficiently represent and reason about joint multivariate distributions. We use graph decomposition to derive an exact, compact, convex formulation for large-margin estimation of Markov networks with sequence and other low-treewidth structure. Seamless integration of kernels with graphical models allows efficient, accurate prediction in real-world tasks. We analyze the theoretical generalization properties of max-margin estimation in Markov networks and derive a novel type of bound on structured error. Using an efficient online-style algorithm that exploits inference in the model and analytic updates, we solve very large estimation problems.

We define an important subclass of Markov networks, associative Markov networks (AMNs), which captures positive correlations between variables and permits exact inference which scales up to tens of millions of nodes and edges. While likelihood-based methods are believed to be intractable for AMNs over binary variables, our framework allows exact estimation of such networks of arbitrary connectivity and topology. We also introduce relational Markov networks (RMNs), which compactly define templates for Markov networks for domains with relational structure: objects, attributes, relations.

In addition to graphical models, our framework applies to a wide range of other models: We exploit context free grammar structure to derive a compact max-margin formulation that allows high-accuracy parsing in cubic time by using novel kinds of lexical information. We use combinatorial properties of weighted matchings to develop an exact, efficient formulation for learning to match and apply it to prediction of disulfide connectivity in proteins. Finally, we derive a max-margin formulation for learning the scoring metric for clustering from clustered training data, which tightly integrates metric learning with the clustering algorithm, tuning one to the other in a joint optimization.

We describe experimental applications to a diverse range of tasks, including handwriting recognition, 3D terrain classification, disulfide connectivity prediction in proteins, hypertext categorization, natural language parsing, email organization and image segmentation. These empirical evaluations show significant improvements over state-of-the-art methods and promise wide practical use for our framework.

Acknowledgements

I am profoundly grateful to my advisor, Daphne Koller. Her tireless pursuit of excellence in research, teaching, advising, and every other aspect of her academic work is truly inspirational. I am indebted to Daphne for priceless and copious advice about selecting interesting problems, making progress on difficult ones, pushing ideas to their full development, writing and presenting results in an engaging manner.

I would like to thank my thesis committee, Andrew Ng and Fernando Pereira, as well as my defense committee members, Stephen Boyd and Sebastian Thrun, for their excellent suggestions and thought-provoking questions. I have learned a great deal from their work and their influence on this thesis is immense. In particular, Fernando's work on Markov networks has inspired my focus on this subject. Andrew's research on clustering and classification has informed many of the practical problems addressed in the thesis. Sebastian introduced me to the fascinating problems in 3D vision and robotics. Finally, Stephen's book on convex optimization is the primary source of many of the insights and derivations in this thesis.

Daphne's research group is an institution in itself. I am very lucky to have been a part of it and shared the company, the ideas, the questions and the expertise of Pieter Abbeel, Drago Anguelov, Alexis Battle, Luke Biewald, Xavier Boyen, Vassil Chatalbashev, Gal Chechik, Lise Getoor, Carlos Guestrin, Uri Lerner, Uri Nodelman, Dirk Ormoneit, Ron Parr, Eran Segal, Christian Shelton, Simon Tong, David Vickrey, Haidong Wang and Ming Fai Wong.

Jean-Claude Latombe was my first research advisor when I was still an undergraduate at Stanford. His thoughtful guidance and quick wit were a great welcome and inspiration to continue my studies. When I started my research in machine learning with Daphne, I

had the privilege and pleasure to work with Nir Friedman, Lise Getoor and Eran Segal. They have taught me to remain steadfast when elegant theories meet the ragged edge of real problems. Lise has been a great friend and my other big sister throughout my graduate life. I could not wish for more caring advice and encouragement than she has given me all these years.

I was lucky to meet Michael Collins, Michael Littman and David McAllester all in one place, while I was a summer intern at AT&T. Collins' work on learning large-margin tagging and parsing models motivated me to look at such methods for other structured models. I am very glad I had the chance to work with him on the max-margin parsing project. Michael Littman, with his endearing combination of humor and warmth, lightness and depth, has been a constant source of encouragement and inspiration. David's wide-ranging research on probability and logic, languages and generalization bounds, to mention a few, enlightened much of my work.

(Almost) everything I know about natural language processing I learned from Chris Manning and Dan Klein. Chris' healthy scepticism and emphasis on the practical has kept me from going down many dead-end paths. Dan has an incredible gift of making ideas work, and better yet, explaining why other ideas do not work.

I would like to express my appreciation and gratitude to all my collaborators for their excellent ideas, hard work and dedication: Pieter Abbeel, Drago Anguelov, Peter Bartlett, Luke Biewald, Vassil Chatalbashev, Michael Collins, Nir Friedman, Audrey Gasch, Lise Getoor, Carlos Guestrin, Geremy Heitz, Dan Klein, Daphne Koller, Chris Manning, David McAllester, Eran Segal, David Vickrey and Ming Fai Wong.

Merci beaucoup to Pieter Abbeel, my sounding board for many ideas in convex optimization and learning, for numerous enlightening discussions that we have had anytime, anywhere: gym, tennis court, biking to EV, over cheap Thai food. Ogromnoye spasibo to my officemate, sportsmate, always late, ever-ready to checkmate, Drago Anguelov, who is always willing to share with me everything from research problems to music and backpacking treks to Russian books and Belgian beer. Blagodarya vi mnogo for another great Bulgarian I have the good luck to know, Vasco Chatalbashev, for his company and ideas, good cheer and the great knack for always coming through. Muchas gracias to Carlos Guestrin, my personal climbing instructor, fellow photography fanatic, avid grillmaster

and exploiter-of-structure extraordinaire, for opening his ear and heart to me when it matters most. Mh-goi-saai to Ming Fai Wong, for his sense of humor, kind heart and excellent work.

Many thanks to my friends who have had nothing to do with work in this thesis, but worked hard to keep my relative sanity throughout. I will not list all of you here, but my gratitude to you is immense.

My parents, Mark and Tsilya, have given me unbending support and constant encouragement. I thank them for all the sacrifices they have made to ensure that their children would have freedom and opportunities they never had in Soviet Union. To my sister, Anna, my brother-in-law, Ilya, and my sweet niece, Talia, I am grateful for bringing me so much joy and love. To my tireless partner in work and play, tears and laughter, hardship and love, Anat Caspi, thank you for making me happy next to you.

Contents

Abstract **vii**

Acknowledgements **ix**

1 Introduction **1**

1.1 Supervised learning 2
1.2 Complex prediction problems 4
1.3 Structured models 6
1.4 Contributions 8
1.5 Thesis outline 11
1.6 Previously published work 13

I Models and methods **15**

2 Supervised learning **16**

2.1 Classification with generalized linear models 19
2.2 Logistic regression 20
2.3 Logistic dual and maximum entropy 21
2.4 Support vector machines 21
2.5 SVM dual and kernels 22

3 Structured models **24**

3.1 Probabilistic models: generative and conditional 25

3.2 Prediction models: normalized and unnormalized 27
3.3 Markov networks 27
3.3.1 Representation 28
3.3.2 Inference 31
3.3.3 Linear programming MAP inference 34
3.4 Context free grammars 37
3.5 Combinatorial problems 40

4 Structured maximum margin estimation 42
4.1 Max-margin estimation 43
4.1.1 Min-max formulation 43
4.1.2 Certificate formulation 48
4.2 Approximations: upper and lower bounds 50
4.2.1 Constraint generation 50
4.2.2 Constraint strengthening 52
4.3 Related work 53
4.4 Conclusion 55

II Markov networks 57

5 Markov networks 58
5.1 Maximum likelihood estimation 59
5.2 Maximum margin estimation 61
5.3 M^3N dual and kernels 65
5.4 Untriangulated models 69
5.5 Generalization bound 70
5.6 Related work 73
5.7 Conclusion 74

6 M^3N algorithms and experiments 75
6.1 Solving the M^3N QP 75
6.1.1 SMO 78

6.1.2 Selecting SMO pairs 79
6.1.3 Structured SMO 81
6.2 Experiments 85
6.3 Related work 87
6.4 Conclusion 87

7 Associative Markov networks 89
7.1 Associative networks 90
7.2 LP Inference 91
7.3 Min-cut inference 93
7.3.1 Graph construction 93
7.3.2 Multi-class case 95
7.4 Max-margin estimation 97
7.5 Experiments 99
7.6 Related work 104
7.7 Conclusion 104

8 Relational Markov networks 109
8.1 Relational classification 110
8.2 Relational Markov networks 113
8.3 Approximate inference and learning 116
8.4 Experiments 118
8.4.1 Flat models 119
8.4.2 Link model 120
8.4.3 Cocite model 121
8.5 Related work 123
8.6 Conclusion 124

III Broader applications: parsing, matching, clustering 127

9 Context free grammars 128
9.1 Context free grammar model 128

9.2 Context free parsing 132
9.3 Discriminative parsing models 133
9.3.1 Maximum likelihood estimation 134
9.3.2 Maximum margin estimation 135
9.4 Structured SMO for CFGs 138
9.5 Experiments 138
9.6 Related work 142
9.7 Conclusion 143

10 Matchings **144**
10.1 Disulfide connectivity prediction 145
10.2 Learning to match 146
10.3 Min-max formulation 148
10.4 Certificate formulation 150
10.5 Kernels 153
10.6 Experiments 154
10.7 Related work 157
10.8 Conclusion 159

11 Correlation clustering **160**
11.1 Clustering formulation 161
11.1.1 Linear programming relaxation 162
11.1.2 Semidefinite programming relaxation 163
11.2 Learning formulation 164
11.3 Dual formulation and kernels 167
11.4 Experiments 168
11.4.1 Irrelevant features 168
11.4.2 Email clustering 168
11.4.3 Image segmentation 170
11.5 Related work 172
11.6 Conclusion 173

IV Conclusions and future directions **175**

12 Conclusions and future directions **176**

12.1 Summary of contributions 176

12.1.1 Structured maximum margin estimation 177

12.1.2 Markov networks: max-margin, associative, relational 180

12.1.3 Broader applications: parsing, matching, clustering 182

12.2 Extensions and open problems 183

12.2.1 Theoretical analysis and optimization algorithms 183

12.2.2 Novel prediction tasks 184

12.2.3 More general learning settings 185

12.3 Future 187

A Proofs and derivations **188**

A.1 Proof of Theorem 5.5.1 188

A.1.1 Binary classification 189

A.1.2 Structured classification 190

A.2 AMN proofs and derivations 195

A.2.1 Binary AMNs 195

A.2.2 Multi-class AMNs 197

A.2.3 Derivation of the factored primal and dual max-margin QP 199

Bibliography **202**

List of Tables

9.1 Parsing results on development set 140
9.2 Parsing results on test set 140

10.1 Bond connectivity prediction results 156

List of Figures

1.1 Supervised learning setting 3
1.2 Examples of complex prediction problems 4

2.1 Handwritten character recognition 19
2.2 Classification loss and upper bounds 20

3.1 Handwritten word recognition 25
3.2 Chain Markov network 29
3.3 Diamond Markov network 32
3.4 Diamond network junction tree 34
3.5 Marginal agreement 35
3.6 Example parse tree 37

4.1 Exact and approximate constraints for max-margin estimation 51
4.2 A constraint generation algorithm. 52

5.1 Chain M^3N example 64
5.2 Diamond Markov network 69

6.1 Block-coordinate ascent 77
6.2 SMO subproblem 79
6.3 SMO pair selection 80
6.4 Structured SMO diagram 82
6.5 Structured SMO pair selection 85
6.6 OCR results 86

7.1 Min-cut graph construction 94
7.2 α-expansion algorithm 96
7.3 Segbot: roving robot equipped with SICK2 laser sensors. 100
7.4 3D laser scan range map of the Stanford Quad. 101
7.5 Comparison of terrain classification models (detail) 102
7.6 Min-cut inference running times 103
7.7 Comparison of terrain classification models (zoomed-out) 106
7.8 Labeled portion of the test terrain dataset (Ground truth and SVM predictions) 107
7.9 Labeled portion of the test terrain dataset (Voted-SVM and AMN predictions) 108

8.1 Link model for document classification 113
8.2 WebKB results: flat models 120
8.3 WebKB results: link model 121
8.4 WebKB results: cocite model 122

9.1 Two representations of a binary parse tree 130

10.1 PDB protein 1ANS 147
10.2 Number of constraints vs. number of bonds 150
10.3 Learning curve for bond connectivity prediction 157

11.1 Learning to cluster with irrelevant features 169
11.2 Learning to organize email 170
11.3 Two segmentations by different users 171
11.4 Two segmentations by different models 172

Chapter 1

Introduction

The breadth of tasks addressed by machine learning is rapidly expanding. Major applications include medical diagnosis, scientific discovery, financial analysis, fraud detection, DNA sequence analysis, speech and handwriting recognition, game playing, image analysis, robot locomotion and many more. Of course, the list of things we would like a computer to learn to do is much, much longer. As we work our way down that list, we encounter the need for very sophisticated decision making from our programs.

Some tasks, for example, handwriting recognition, are performed almost effortlessly by a person, but remain difficult and error-prone for computers. The complex synthesis of many levels of signal processing a person executes when confronted by a line of handwritten text is daunting. The reconstruction of an entire sentence from the photons hitting the retina off of each tiny patch of an image undoubtedly requires an elaborate interplay of recognition and representation of the pen-strokes, the individual letters, whole words and constituent phrases.

Computer scientists, as opposed to, say, neuroscientists, are primarily concerned with achieving acceptable speed and accuracy of recognition rather than modeling this complicated process with any biological verity. Computational models for handwriting recognition aim to capture the salient properties of the problem: typical shapes of the letters, likely letter combinations that make up words, common ways to combine words into phrases, frequent grammatical constructions of the phrases, etc. Machine learning offers an alternative to encoding all the intricate details of such a model from scratch. One of its primary goals

is to devise efficient algorithms for training computers to automatically acquire effective and accurate models from experience.

In this thesis, we present a discriminative learning framework and a novel family of efficient models and algorithms for complex recognition tasks in several disciplines, including natural language processing, computer vision and computational biology. We develop theoretical foundations for our approach and show a wide range of experimental applications, including handwriting recognition, 3-dimensional terrain classification, disulfide connectivity in protein structure prediction, hypertext categorization, natural language parsing, email organization and image segmentation.

1.1 Supervised learning

The most basic supervised learning task is classification. Suppose we wish to learn to recognize a handwritten character from a scanned image. This is a classification task, because we must assigns a class (an English letter from 'a' through 'z') to an observation of an object (an image). Essentially, a classifier is a function that maps an input (an image) to an output (a letter). In the supervised learning setting, we construct a classifier by observing labeled training examples, in our case, sample images paired with appropriate letters. The main problem addressed by supervised learning is generalization. The learning program is allowed to observe only a small sample of labeled images to produce an accurate classifier on unseen images of letters.

More formally, let $\mathbf{x}$ denote an input. For example, a black-and-white image $\mathbf{x}$ can be represented as a vector of pixel intensities. We use $\mathcal{X}$ to denote the space of all possible inputs. Let y denote the output, and $\mathcal{Y}$ be the discrete space of possible outcomes (e.g., 26 letters 'a'-'z'). A classifier (or hypothesis) h is a function from $\mathcal{X}$ to $\mathcal{Y}$, $h : \mathcal{X} \mapsto \mathcal{Y}$. We denote the set of all classifiers that our learning program can produce as $\mathcal{H}$ (hypothesis class). Then given a set of labeled examples $\{\mathbf{x}^{(i)}, y^{(i)}\}$, $i = 1, \ldots, m$, a learning program seeks to produce a classifier $h \in \mathcal{H}$ that will work well on unseen examples $\mathbf{x}$, usually by finding h that accurately classifies training data. The diagram in Fig. 1.1 summarizes the supervised learning setting.

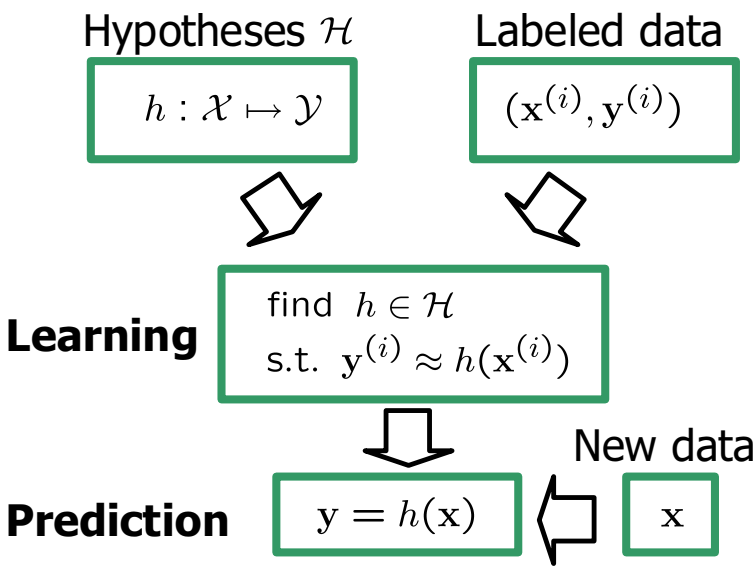
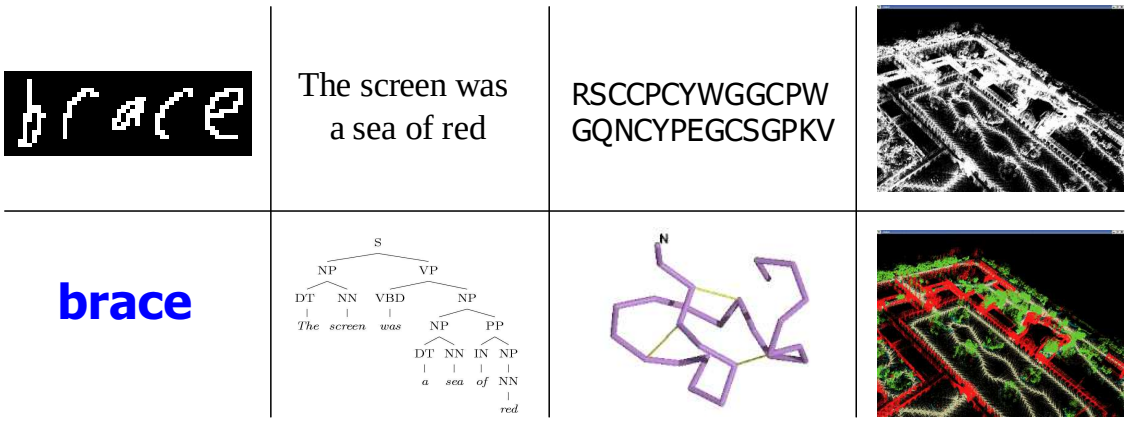


Figure 1.1: Supervised learning setting

The problem of classification has a long history and highly developed theory and practice (see for example, Mitchell [1997]; Vapnik [1995]; Duda *et al.* [2000]; Hastie *et al.* [2001]). The two most important dimensions of variation of classification algorithms is the hypothesis class $\mathcal{H}$ and the criterion for selection of a hypothesis h from $\mathcal{H}$ given the training data. In this thesis, we build upon the generalized linear model family, which underlies standard classifiers such as logistic regression and support vector machines. Through the use of kernels to implicitly define high-dimensional and even infinite-dimensional input representations, generalized linear models can approximate arbitrarily complex decision boundaries.

The task of selecting a hypothesis h reduces to estimating model parameters. Broadly speaking, probabilistic estimation methods associate a joint distribution $p(\mathbf{x}, \mathbf{y})$ or conditional distribution $p(\mathbf{y} \mid \mathbf{x})$ with h and select a model based on the likelihood of the data [Hastie *et al.*, 2001]. Joint distribution models are often called generative, while conditional models are called discriminative. Large margin methods, by contrast, select a model based on a more direct measure of confidence of its predictions on the training data called the margin [Vapnik, 1995]. The difference between these two methods is one of the key



(a) (b) (c) (d)

Figure 1.2: Examples of complex prediction problems (inputs-top, outputs-bottom):
(a) handwriting recognition [image $\mapsto$ word];
(b) natural language parsing [sentence $\mapsto$ parse tree];
(c) disulfide bond prediction in proteins [amino-acid sequence $\mapsto$ bond structure (shown in yellow)];
(d) terrain segmentation [3D image $\mapsto$ segmented objects (trees, bushes, buildings, ground)]

themes in this thesis.

Most of the research has focused on the analysis and classification algorithms for the case of binary outcomes $|\mathcal{Y}| = 2$, or a small number of classes. In this work, we focus on prediction tasks that involve not a single decision with a small set of outcomes, but a complex, interrelated collection of decisions.

1.2 Complex prediction problems

Consider once more the problem of character recognition. In fact, a more natural and useful task is recognizing words and entire sentences. Fig. 1.2(a) shows an example handwritten word "brace." Distinguishing between the second letter and fourth letter ('r' and 'c') *in isolation* is actually far from trivial, but in the context of the surrounding letters that together form a word, this task is much less error-prone for humans and should be for computers as well. It is also more complicated, as different decisions must be weighed against each other to arrive at the globally satisfactory prediction. The space of all possible outcomes

$\mathcal{Y}$ is immense, usually exponential in the number of individual decisions, for example, the number of 5 letter sequences (26^5). However, most of these outcomes are unlikely given the observed input. By capturing the most salient structure of the problem, for example the strong local correlations between consecutive letters, we will construct compact models that efficiently deal with this complexity. Below we list several examples from different fields.

- **Natural language processing**
  Vast amounts of electronically available text have spurred a tremendous amount of research into automatic analysis and processing of natural language. We mention some of the lower-level tasks that have received a lot of recent attention [Charniak, 1993; Manning & Schütze, 1999]. Part-of-speech tagging involves assigning each word in a sentence a part-of-speech tag, such as *noun*, *verb*, *pronoun*, etc. As with handwriting recognition, capturing sequential structure of correlations between consecutive tags is key. In parsing, the goal is to recognize the recursive phrase structure of a sentence, such as verbal, noun and prepositional phrases and their nesting in relation to each other. Fig. 1.2(b) shows a parse tree corresponding to the sentence: "The screen was a sea of red" (more on this in Ch. 9). Many other problems, such as named-entity and relation extraction, text summarization, translation, involve complex global decision making.

- **Computational biology**
  The last two decades have yielded a wealth of high-throughput experimental data, including complete sequencing of many genomes, precise measurements of protein 3D structure, genome-wide assays of mRNA levels and protein-protein interactions. Major research has been devoted to gene-finding, alignment of sequences, protein structure prediction, molecular pathway discovery [Gusfield, 1997; Durbin *et al.*, 1998]. Fig. 1.2(c) shows disulfide bond structure (shown in yellow) we would like to predict from the amino-acid sequence of the protein (more on this in Ch. 10).

- **Computer vision**
  As digital cameras and optical scanners become commonplace accessories, medical imaging technology produces detailed physiological measurements, laser scanners

capture 3D environments, satellites and telescopes bring pictures of Earth and distant stars, we are flooded with images we would like our computer to analyze. Example tasks include object detection and segmentation, motion tracking, 3D reconstruction from stereo and video, and much more [Forsyth & Ponce, 2002]. Fig. 1.2(d) shows a 3D laser range data image of the Stanford campus collected by a roving robot which we would like to segment into objects such as trees, bushes, buildings, ground, etc. (more on this in Ch. 7).

1.3 Structured models

This wide range of problems have been tackled using various models and methods. We focus on the models that compactly capture correlation and constraint structure inherent to many tasks. Abstractly, a model assigns a score (or likelihood in probabilistic models) to each possible input/output pair $(\mathbf{x}, \mathbf{y})$, typically through a compact, parameterized scoring function. Inference in these models refers to computing the highest scoring output given the input and usually involves dynamic programming or combinatorial optimization.

- **Markov networks**
  Markov networks (a.k.a. Markov random fields) are extensively used to model complex sequential, spatial, and relational interactions in prediction problems arising in many fields. These problems involve labeling a set of related objects that exhibit *local* consistency. Markov networks compactly represent complex joint distributions of the label variables by modeling their local interactions. Such models are encoded by a graph, whose nodes represent the different object labels, and whose edges represent and quantify direct dependencies between them. The graphical structure of the models encodes the *qualitative* aspects of the distribution: direct dependencies as well as conditional independencies. The *quantitative* aspect of the model is defined by the *potentials* that are associated with nodes and cliques of the graph. The graphical structure of the network (more precisely, the treewidth of the graph, which we formally define in Ch. 3) is critical to efficient inference and learning in the model.
- **Context free grammars**

Context-free grammars are one of the primary formalisms for capturing the recursive structure of syntactic constructions [Manning & Schütze, 1999]. For example, in Fig. 1.2, the non-terminal symbols (labels of internal nodes) correspond to syntactic categories such as noun phrase (NP), verbal phrase (VP) or prepositional phrase (PP) and part-of-speech tags like nouns (NN), verbs (VBD), determiners (DT) and prepositions (IN). The terminal symbols (leaves) are the words of the sentence. A CFG consists of recursive productions (e.g. $VP \rightarrow VP\ PP$, $\mathrm{DT} \rightarrow \mathrm{The}$) that can be applied to derive a sentence of the language. The productions define the set of syntactically allowed phrase structures (derivations). By compactly defining a probability distribution over individual productions, probabilistic CFGs construct a distribution over parse trees and sentences, and the prediction task reduces to finding the most likely tree given the sentence. The context free restriction allows efficient inference and learning in such models.

- **Combinatorial structures**
  Many important computational tasks are formulated as combinatorial optimization problems such as the maximum weight bipartite and perfect matching, spanning tree, graph-cut, edge-cover, and many others [Lawler, 1976; Papadimitriou & Steiglitz, 1982; Cormen *et al.*, 2001]. Although the term 'model' is often reserved for probabilistic models, we use the term model very broadly, to include any scheme that assigns scores to the output space $\mathcal{Y}$ and has a procedure for finding the optimal scoring $\mathbf{y}$. For example, the disulfide connectivity prediction in Fig. 1.2(c) can be modeled by maximum weight perfect matchings, where the weights define potential bond strength based on the local amino-acid sequence properties. The other combinatorial structures we consider and apply in this thesis include graph cuts and partitions, bipartite matchings, and spanning trees.

The standard methods of estimation for Markov networks and context free grammars are based on maximum likelihood, both joint and conditional. However, maximum likelihood estimation of scoring function parameters for combinatorial structures is often intractable because of the problem of defining a normalized distribution over an exponential set of combinatorial structures.

1.4 Contributions

This thesis addresses the problem of efficient learning of high-accuracy models for complex prediction problems. We consider a very large class of structured models, from Markov networks to context free grammars to combinatorial graph structures such as matchings and cuts. We focus on those models where exact inference is tractable, or can be efficiently approximated.

- **Learning framework for structured models**
  We propose a general framework for efficient estimation of models for structured prediction. An alternative to likelihood-based methods, this framework builds upon the large margin estimation principle. Intuitively, we find parameters such that inference in the model (dynamic programming, combinatorial optimization) predicts the correct answers on the training data with maximum confidence. We develop general conditions under which exact large margin estimation is tractable and present two formulations for structured max-margin estimation that define compact convex optimization problems, taking advantage of prediction task structure. The first formulation relies on the ability to express inference in the model as a compact convex optimization problem. The second one only requires compactly expressing optimality of a given assignment according to the model and applies to a broader range of combinatorial problems. These two formulations form the foundation which the rest of the thesis develops.

- **Markov networks**
  The largest portion of the thesis is devoted to novel estimation algorithms, representational extensions, generalization analysis and experimental validation for Markov networks, a model class of choice in many structured prediction tasks in language, vision and biology.

  - **Low-treewidth Markov networks**
    We use graph decomposition to derive an exact, compact, convex learning formulation for Markov networks with sequence and other low-treewidth structure. The seamless integration of kernels with graphical models allows us to create

very rich models that leverage the immense amount of research in kernel design and graphical model decompositions for efficient, accurate prediction in real-world tasks. We also use approximate graph decomposition to derive a compact approximate formulation for Markov networks in which inference is intractable.

▷ **Scalable online algorithm**
We present an efficient algorithm for solving the estimation problem called Structured SMO. Our online-style algorithm uses inference in the model and analytic updates to solve extremely large estimation problems.

▷ **Generalization analysis**
We analyze the theoretical generalization properties of max-margin estimation in Markov networks and derive a novel margin-based bound for structured prediction. This bound is the first to address structured error (e.g. proportion of mislabeled pixels in an image) and uses a proof that exploits the graphical model structure.

▷ **Learning associative Markov networks (AMNs)**
We define an important subclass of Markov networks that captures positive correlations present in many domains. We show that for AMNs over binary variables, our framework allows exact estimation of networks of arbitrary connectivity and topology, for which likelihood methods are believed to be intractable. For the non-binary case, we provide an approximation that works well in practice. We present an AMN-based method for object segmentation from 3D range data. By constraining the class of Markov networks to AMNs, our models are learned efficiently and, at run-time, scale up to tens of millions of nodes and edges.

▷ **Representation and learning of relational Markov networks**
We introduce relational Markov networks (RMNs), which compactly define templates for Markov networks for domains with relational structure objects,

attributes, relations. The graphical structure of an RMN is based on the relational structure of the domain, and can easily model complex interaction patterns over related entities. We use approximate inference in these complex models, in which exact inference is intractable, to derive an approximate learning formulation. We apply this class of models to classification of hypertext using hyperlink structure to define relations between webpages.

- **Broader applications: parsing, matching, clustering**
  The other large portion the thesis addresses a range of prediction tasks with very diverse models: context free grammars for natural language parsing, perfect matchings for disulfide connectivity in protein structure prediction, graph partitions for clustering documents and segmenting images.

  - **Learning to parse**
    We exploit context free grammar structure to derive a compact max-margin formulation and show high-accuracy parsing in cubic time by exploiting novel kinds of lexical information. We show experimental evidence of the model's improved performance over several baseline models.

  - **Learning to match**
    We use combinatorial properties of weighted matchings to develop an exact, efficient algorithm for learning to match. We apply our framework to prediction of disulfide connectivity in proteins using perfect non-bipartite matchings. The algorithm we propose uses kernels, which makes it possible to efficiently embed the features in very high-dimensional spaces and achieve state-of-the-art accuracy.

  - **Learning to cluster**
    We derive a max-margin formulation for learning the affinity metric for clustering from clustered training data. In contrast to algorithms that learn a metric independently of the algorithm that will be used to cluster the data, we describe a formulation that tightly integrates metric learning with the clustering algorithm, tuning one to the other in a joint optimization. Experiments on synthetic and real-world data show the ability of the algorithm to learn an appropriate

clustering metric for a variety of desired clusterings, including email folder organization and image segmentation.

1.5 Thesis outline

Below is a summary of the rest of the chapters in the thesis:

Chapter 2. Supervised learning: We review basic definitions and statistical framework for classification. We define hypothesis classes, loss functions, risk. We consider generalized linear models, including logistic regression and support vector machines, and review estimation methods based on maximizing likelihood, conditional likelihood and margin. We describe the relationship between the dual estimation problems and kernels.

Chapter 3. Structured models: In this chapter, we define the abstract class of structured prediction problems and models addressed by the thesis. We compare probabilistic models, generative and discriminative and unnormalized models. We describe representation and inference for Markov networks, including dynamic and linear programming inference. We also briefly describe context free grammars and combinatorial structures as models.

Chapter 4. Structured maximum margin estimation: This chapter outlines the main principles of maximum margin estimation for structured models. We address the exponential blow-up of the naive problem formulation by deriving two general equivalent convex formulation. These formulations, min-max and certificate, allow us to exploit decomposition and combinatorial structure of the prediction task. They lead to polynomial size programs for estimation of models where the prediction problem is tractable. We also discuss approximations, in particular using upper and lower bounds, for solving intractable or very large problems.

Chapter 5. Markov networks: We review maximum conditional likelihood estimation and present maximum margin estimation for Markov networks. We use graphical model decomposition to derive a convex, compact formulation that seamlessly integrates kernels with graphical models. We analyze the theoretical generalization

properties of max-margin estimation and derive a novel margin-based bound for structured classification.

Chapter 6. M^{3}N algorithms and experiments: We present an efficient algorithm for solving the estimation problem in graphical models, called Structured SMO. Our online-style algorithm uses inference in the model and analytic updates to solve extremely large quadratic problems. We present experiments with handwriting recognition, where our models significantly outperform other approaches by effectively capturing correlation between adjacent letters and incorporating high-dimensional input representation via kernels.

Chapter 7. Associative Markov networks: We define an important subclass of Markov networks, associative Markov networks (AMNs), that captures positive interactions present in many domains. We show that for associative Markov networks of over binary variables, max-margin estimation allows exact training of networks of arbitrary connectivity and topology, for which maximum likelihood methods are believed to be intractable. For the non-binary case, we provide an approximation that works well in practice. We present an AMN-based method for object segmentation from 3D range data that scales to very large prediction tasks involving tens of millions of points.

Chapter 8. Relational Markov networks: We introduce the framework of relational Markov networks (RMNs), which compactly defines templates for Markov networks in domains with rich structure modeled by objects, attributes and relations. The graphical structure of an RMN is based on the relational structure of the domain, and can easily model complex patterns over related entities. As we show, the use of an undirected, discriminative graphical model avoids the difficulties of defining a coherent generative model for graph structures in directed models and allows us tremendous flexibility in representing complex patterns. We provide experimental results on a webpage classification task, showing that accuracy can be significantly improved by modeling relational dependencies.

Chapter 9. Context free grammars: We present max-margin estimation for natural language parsing on the decomposition properties of context free grammars. We show

that this framework allows high-accuracy parsing in cubic time by exploiting novel kinds of lexical information. We show experimental evidence of the model's improved performance over several baseline models.

Chapter 10. Perfect matchings: We apply our framework to learning to predict disulfide connectivity in proteins using perfect matchings. We use combinatorial properties of weighted matchings to develop an exact, efficient algorithm for learning the parameters of the model. The algorithm we propose uses kernels, which makes it possible to efficiently embed the features in very high-dimensional spaces and achieve state-of-the-art accuracy.

Chapter 11. Correlation clustering: In this chapter, we derive a max-margin formulation for learning affinity scores for correlation clustering from clustered training data. We formulate the approximate learning problem as a compact convex program with quadratic objective and linear or positive-semidefinite constraints. Experiments on synthetic and real-world data show the ability of the algorithm to learn an appropriate clustering metric for a variety of desired clusterings, including email folder organization and image segmentation.

Chapter 12. Conclusions and future directions: We review the main contributions of the thesis and summarize their significance, applicability and limitations. We discuss extensions and future research directions not addressed in the thesis.

1.6 Previously published work

Some of the work described in this thesis has been published in conference proceedings. The min-max and certificate formulations for structured max-margin estimation have not been published in their general form outlined in Ch. 4, although they underly several papers mentioned below. The polynomial formulation of maximum margin Markov networks presented in Ch. 5 was published for a less general case, using a dual decomposition technique [Taskar *et al.*, 2003a]. Work on associative Markov networks (Ch. 7) was published with experiments on hypertext and news-wire classification [Taskar *et al.*, 2004a]. A paper

on 3D object segmentation using AMNs, which presents a experiments on terrain classification and other tasks, is currently under review (joint work with Drago Anguelov, Vassil Chatalbashev, Dinkar Gupta, Geremy Heitz, Daphne Koller and Andrew Ng). Taskar *et al.* [2002] and Taskar *et al.* [2003b] defined and applied the Relational Markov networks (Ch. 8), using maximum (conditional) likelihood estimation. Natural language parsing in Ch. 9 was published in Taskar *et al.* [2004b]. Disulfide connectivity prediction using perfect matchings in Ch. 10 (joint work with Vassil Chatalbashev and Daphne Koller) is currently under review. Finally, work on correlation clustering in Ch. 11, done jointly with Pieter Abbeel and Andrew Ng, has not been published.

Part I

Models and methods

Chapter 2

Supervised learning

In supervised learning, we seek a function $h : \mathcal{X} \mapsto \mathcal{Y}$ that maps inputs $\mathbf{x} \in \mathcal{X}$ to outputs $y \in \mathcal{Y}$. The input space $\mathcal{X}$ is an arbitrary set (often $\mathcal{X} = \mathbb{R}^n$), while the output space $\mathcal{Y}$ we consider in this chapter discrete. A supervised learning problem with discrete outputs, $\mathcal{Y} = \{y_1, \ldots, y_k\}$, where k is the number of classes, is called **classification**. In handwritten character recognition, for example, $\mathcal{X}$ is the set of images of letters and $\mathcal{Y}$ is the alphabet (see Fig. 2.1).

The input to an algorithm is **training data**, a set of m i.i.d. (independent and identically distributed) samples $S = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$ drawn from a fixed but unknown distribution D over $\mathcal{X} \times \mathcal{Y}$. The goal of a learning algorithm is to output a hypothesis h such that $h(\mathbf{x})$ will approximate y on new samples from the distribution $(\mathbf{x}, y) \sim D$.

Learning algorithms can be distinguished among several dimensions, chief among them is the **hypothesis class** $\mathcal{H}$ of functions h the algorithm outputs. Numerous classes of functions have been well studied, including decision trees, neural networks, nearest-neighbors, generalized log-linear models and kernel methods (see Quinlan [2001]; Bishop [1995]; Hastie *et al.* [2001]; Duda *et al.* [2000], for in-depth discussion of these and many other models). We will concentrate on the last two classes, for several reasons we discuss below, including accuracy, efficiency, and extensibility to more complex structured prediction tasks will consider in the next chapter.

The second crucial dimension of a learning algorithm is the criterion for selection of h from $\mathcal{H}$. We arrive at such a criterion by quantifying what it means for $h(\mathbf{x})$ to approximate

y. The **risk functional** $\mathcal{R}_D^\ell[(h)]$ measures the expected error of the approximation:

$$\mathcal{R}_D^\ell[h] = \mathbf{E}_{(\mathbf{x},y)\sim D}[\ell(\mathbf{x}, y, h(\mathbf{x}))], \tag{2.1}$$

where the **loss function** $\ell : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ measures the penalty for predicting $h(\mathbf{x})$ on the sample $(\mathbf{x}, y)$. In general, we assume that $\ell(\mathbf{x}, y, \hat{y}) = 0$ if $y = \hat{y}$.

A common loss function for classification is $0/1$ **loss**

$$\ell^{0/1}(\mathbf{x}, y, h(\mathbf{x})) \equiv \mathbb{1}(y \neq h(\mathbf{x})),$$

where $\mathbb{1}(\cdot)$ denotes the indicator function, that is, $\mathbb{1}(\mathrm{true}) = 1$ and $\mathbb{1}(\mathrm{false}) = 0$.

Since we do not generally know the distribution D, we estimate the risk of h using its **empirical risk** $\mathcal{R}_S^\ell$, computed on the training sample S:

$$\mathcal{R}_S^\ell[h] = \frac{1}{m}\sum_{i=1}^{m} \ell(\mathbf{x}^{(i)}, y^{(i)}, h(\mathbf{x}^{(i)})) = \frac{1}{m}\sum_{i=1}^{m} \ell_i(h(\mathbf{x}^{(i)})), \tag{2.2}$$

where we abbreviate $\ell(\mathbf{x}^{(i)}, y^{(i)}, h(\mathbf{x}^{(i)})) = \ell_i(h(\mathbf{x}^{(i)}))$. For $0/1$ loss, $\mathcal{R}_S^\ell[h]$ is simply the proportion of training examples that h misclassifies. $\mathcal{R}_S^\ell[h]$ is often called the **training error** or **training loss**.

If our set of hypotheses, $\mathcal{H}$, is large enough, we will be able to find h that has zero or very small empirical risk. However, simply selecting a hypothesis with lowest risk

$$h^* = \arg\min_{h\in\mathcal{H}} \mathcal{R}_S^\ell[h],$$

is generally not a good idea. For example, if $\mathcal{X} = \mathbb{R}, \mathcal{Y} = \mathbb{R}$ and $\mathcal{H}$ includes all polynomials of degree $m-1$, we can always find a polynomial h that passes through all the sample points $(x^{(i)}, y^{(i)})$, $i = (1, ..., m)$ assuming that all the $x^{(i)}$ are unique. This polynomial is very likely to **overfit** the training data, that is, it will have zero empirical risk, but high actual risk. The key to selecting a good hypothesis is to trade-off complexity of class $\mathcal{H}$ (e.g. the degree of the polynomial) with the error on the training data as measured by empirical risk $\mathcal{R}_S^\ell$. For a vast majority of supervised learning algorithms, this fundamental balance is

achieved by minimizing the weighted combination of the two criteria:

$$h^* = \arg\min_{h \in \mathcal{H}} \left(\mathcal{D}[h] + C\mathcal{R}_S^\ell[h] \right), \tag{2.3}$$

where $\mathcal{D}[h]$ measures the inherent dimension or complexity of h, and $C \geq 0$ is a trade-off parameter. We will not go into derivation of various complexity measures $\mathcal{D}[h]$ here, but simply adopt the standard measures as needed and refer the reader to Vapnik [1995]; Devroye *et al.* [1996]; Hastie *et al.* [2001] for details. The term $\mathcal{D}[h]$ is often called **regularization**.

Depending on the complexity of the class $\mathcal{H}$, the search for the optimal h^* in (2.3) may be a daunting task[1]. For many classes, for example decision trees and multi-layer neural networks, it is intractable [Bishop, 1995; Quinlan, 2001], and we must resort to approximate, greedy optimization methods. For these intractable classes, the search procedure used by the learning algorithm is crucial. Below however, we will concentrate on models where the optimal h^* can be found efficiently using convex optimization in polynomial time. Hence, the learning algorithms we consider are completely characterized by the hypothesis class $\mathcal{H}$, the loss function ℓ, and the regularization $\mathcal{D}[h]$.

In general, we consider hypothesis classes of the following parametric form:

$$h_{\mathbf{w}}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} f(\mathbf{w}, \mathbf{x}, y), \tag{2.4}$$

where $f(\mathbf{w}, \mathbf{x}, y)$ is a function $f : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$, where $\mathbf{w} \in \mathcal{W}$ is a set of parameters, usually with $\mathcal{W} \subseteq \mathbb{R}^n$. We assume that ties in the $\arg\max$ are broken using some arbitrary but fixed rule. As we discuss below, this class of hypotheses is very rich and includes many standard models. The formulation in (2.4) of the hypothesis class in terms of an optimization procedure will become crucial to extending supervised learning techniques to cases where the output space $\mathcal{Y}$ is more complex.

[1]For classification, minimizing the objective with the usual $0/1$ training error is generally a very difficult problem with multiple maxima for most realistic $\mathcal{H}$. See discussion in the next section about approaches to dealing with $0/1$ loss.

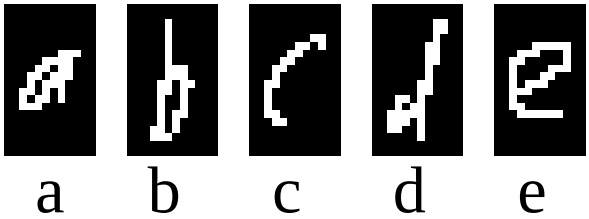


Figure 2.1: Handwritten character recognition: sample letters from Kassel [1995] data set.

2.1 Classification with generalized linear models

For classification, we consider the **generalized linear family** of hypotheses $\mathcal{H}$. Given n real-valued basis functions $f_j : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$, a hypothesis $h_{\mathbf{w}} \in \mathcal{H}$ is defined by a set of n coefficients $w_j \in \mathbb{R}$ such that:

$$h_{\mathbf{w}}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{i=1}^{n} w_j f_j(\mathbf{x}, y) = \arg\max_{y \in \mathcal{Y}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, y). \tag{2.5}$$

Consider the character recognition example in Fig. 2.1. Our input $\mathbf{x}$ is a vector of pixel values of the image and y is the alphabet $\{a, \ldots, z\}$. We might have a basis function $f_j(\mathbf{x}, y) = \mathbb{1}(\mathbf{x}_{row,col} = on \wedge y = char)$ for each possible (row, col) and $char \in \mathcal{Y}$, where $\mathbf{x}_{row,col}$ denotes the value of pixel (row, col). Since different letters tend to have different pixels turned on, this very simple model captures enough information to perform reasonably well.

The most common loss for classification is $0/1$ loss. Minimizing the $0/1$ risk is generally a very difficult problem with multiple maxima for any large class $\mathcal{H}$. The standard solution is minimizing an upper bound on the $0/1$ loss, $\overline{\ell}(\mathbf{x}, y, h(\mathbf{x})) \geq \ell(\mathbf{x}, y, h(\mathbf{x}))$. (In addition to computational advantages of this approach, there are statistical benefits of minimizing a *convex* upper bound [Bartlett *et al.*, 2003]). Two of the primary classification methods we consider, logistic regression and support vector machines, differ primarily in their choice of the upper bound on the training $0/1$ loss. The regularization $\mathcal{D}[h_{\mathbf{w}}]$ for the linear family is typically the norm of the parameters $||\mathbf{w}||_p$ for $p = 1, 2$. Intuitively, a zero, or small weight w_j implies that the hypothesis $h_{\mathbf{w}}$ does not depend on the value of $f_j(\mathbf{x}, y)$ and hence is simpler than a $h_{\mathbf{w}}$ with a large weight w_j.

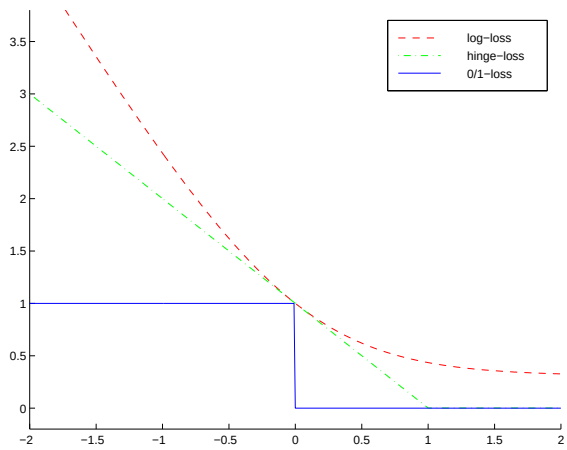


Figure 2.2: $0/1$-loss upper bounded by log-loss and hinge-loss. Horizontal axis shows $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, y) - \max_{y' \neq y} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, y')$, where y is the correct label for $\mathbf{x}$, while the vertical axis show the value of the associated loss. The log-loss is shown up to an additive constant for illustration purposes.

2.2 Logistic regression

In logistic regression, we assign a probabilistic interpretation to the hypothesis $h_{\mathbf{w}}$ as defining a conditional distribution:

$$P_{\mathbf{w}}(y \mid \mathbf{x}) = \frac{1}{Z_{\mathbf{w}}(\mathbf{x})} \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, y)\}, \tag{2.6}$$

where $Z_{\mathbf{w}}(\mathbf{x}) = \sum_{y \in \mathcal{Y}} \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, y)\}$. The optimal weights are selected by maximizing the conditional likelihood of the data (minimizing the log-loss) with some regularization. This approach is called the (regularized) **maximum likelihood estimation**. Common choices for regularization are 1 or 2-norm regularization on the weights; we use 2-norm below:

$$\min \quad \frac{1}{2}||\mathbf{w}||^2 + C \sum_i \log Z_{\mathbf{w}}(\mathbf{x}^{(i)}) - \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, y^{(i)}), \tag{2.7}$$

where C is a user-specified constant the determines the trade-off between regularization and likelihood of the data. The log-loss $\log Z_{\mathbf{w}}(\mathbf{x}) - \mathbf{w}^\top \mathbf{f}(\mathbf{x}, y)$ is an upper bound (up to a constant) on the $0/1$ loss $\ell^{0/1}$ (see Fig. 2.2).

2.3 Logistic dual and maximum entropy

The objective function is convex in the parameters $\mathbf{w}$, so we have an unconstrained (differentiable) convex optimization problem. The gradient with respect to $\mathbf{w}$ is given by:

$$\mathbf{w} + C\sum_i \mathbf{E}_{i,\mathbf{w}}[\mathbf{f}_i(\mathbf{x}^{(i)}, y)] - \mathbf{f}_i(\mathbf{x}^{(i)}, y^{(i)}) = \mathbf{w} - C\sum_i \mathbf{E}_{i,\mathbf{w}}[\Delta\mathbf{f}_i(y)],$$

where $\mathbf{E}_{i,\mathbf{w}}[f(y)] = \sum_y f(y)P_{\mathbf{w}}(y \mid \mathbf{x}^{(i)})$ is the expectation under the conditional distribution $P_{\mathbf{w}}(y \mid \mathbf{x}^{(i)})$ and $\Delta\mathbf{f}_i(\mathbf{y}) = \mathbf{f}(\mathbf{x}^{(i)}, y^{(i)}) - \mathbf{f}(\mathbf{x}^{(i)}, y)$. Ignoring the regularization term, the gradient is zero when the basis function expectations are equal to the basis functions evaluated on the labels $y^{(i)}$. It can be shown [Cover & Thomas, 1991] that the dual of the maximum likelihood problem (without regularization) is the maximum entropy problem:

$$\begin{aligned} \max \quad & -\sum_{i,y} P_{\mathbf{w}}(y \mid \mathbf{x}^{(i)}) \log P_{\mathbf{w}}(y \mid \mathbf{x}^{(i)}) \\ \text{s.t.} \quad & \mathbf{E}_{i,\mathbf{w}}[\Delta\mathbf{f}_i(y)] = 0, \quad \forall i. \end{aligned} \tag{2.8}$$

We can interpret logistic regression as trying to match the empirical basis function expectations while maintaining a high entropy conditional distribution $P_{\mathbf{w}}(y \mid \mathbf{x})$.

2.4 Support vector machines

Support vector machines [Vapnik, 1995] select the weights based on the "margin" of confidence of $h_{\mathbf{w}}$. In the multi-class SVM formulation [Weston & Watkins, 1998; Crammer & Singer, 2001], the margin on example i quantifies by how much the true label "wins" over the wrong ones:

$$\gamma_i = \frac{1}{||\mathbf{w}||}\min_{y \neq y^{(i)}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, y^{(i)}) - \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, y) = \frac{1}{||\mathbf{w}||}\min_{y \neq y^{(i)}} \mathbf{w}^\top \Delta\mathbf{f}_i(y),$$

where $\Delta\mathbf{f}_i(y) = \mathbf{f}(\mathbf{x}^{(i)}, y^{(i)}) - \mathbf{f}(\mathbf{x}^{(i)}, y)$. Maximizing the smallest such margin (and allowing for negative margins) is equivalent to solving the following quadratic program:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \Delta\mathbf{f}_i(y) \geq \ell^{0/1}(y) - \xi_i, \quad \forall i, \ \ \forall y \in \mathcal{Y}. \end{aligned} \tag{2.9}$$

Note that the slack variable ξ_i is constrained to be positive in the above program since $\mathbf{w}^\top \Delta\mathbf{f}_i(y^{(i)}) = 0$ and $\ell^{0/1}(y^{(i)}) = 0$. We can also express ξ_i as $\max_y \ell_i^{0/1}(y) - \mathbf{w}^\top \Delta\mathbf{f}_i(y)$, and the optimization problem Eq. (2.9) in a form similar to Eq. (2.7):

$$\min \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \max_y [\ell_i^{0/1}(y) - \mathbf{w}^\top \Delta\mathbf{f}_i(y)]. \tag{2.10}$$

The hinge-loss $\max_y[\ell_i^{0/1}(y) - \mathbf{w}^\top \Delta\mathbf{f}_i(y)]$ is also an upper bound on the $0/1$ loss $\ell^{0/1}$ (see Fig. 2.2).

2.5 SVM dual and kernels

The form of the dual of Eq. (2.9) is crucial to efficient solution of SVM and the ability to use a high or even infinite dimensional set of basis functions via kernels.

$$\begin{aligned} \max \quad & \sum_{i,y} \alpha_i(y)\ell_i^{0/1}(y) - \frac{1}{2}\left|\left|\sum_{i,y} \alpha_i(y)\Delta\mathbf{f}_i(y)\right|\right|^2 \\ \text{s.t.} \quad & \sum_y \alpha_i(y) = C, \ \forall i; \qquad \alpha_i(y) \geq 0, \ \ \forall i, y. \end{aligned} \tag{2.11}$$

In the dual, the $\alpha_i(y)$ variables correspond to the $\mathbf{w}^\top \Delta\mathbf{f}_i(y) \geq \ell^{0/1}(y) - \xi_i$ constraints in the primal Eq. (2.9). The solution to the dual α^* gives the solution to the primal as a weighted combination of basis functions of examples:

$$\mathbf{w}^* = \sum_{i,y} \alpha_i^*(y)\Delta\mathbf{f}_i(y).$$

The pairings of examples and incorrect labels, (i, y), that have non-zero $\alpha_i^*(y)$, are called **support vectors**.

An important feature of the dual formulation is that the basis functions $\mathbf{f}$ appear only as dot products. Expanding the quadratic term, we have:

$$\left|\left|\sum_{i,y} \alpha_i(y)\Delta\mathbf{f}_i(y)\right|\right|^2 = \sum_{i,y}\sum_{j,\bar{y}} \alpha_i(y)\alpha_j(\bar{y})\Delta\mathbf{f}_i(y)^\top\Delta\mathbf{f}_j(\bar{y}).$$

Hence, as long as the dot product $\mathbf{f}(\mathbf{x}, y)^\top\mathbf{f}(\bar{\mathbf{x}}, \bar{y})$ can be computed efficiently, we can solve Eq. (2.11) independently of the actual dimension of $\mathbf{f}$. Note that at classification time, we also do not need to worry about the dimension of $\mathbf{f}$ since:

$$\mathbf{w}^\top\mathbf{f}(\mathbf{x}, \bar{y}) = \sum_{i,y} \alpha_i(y)\Delta\mathbf{f}_i(y)^\top\mathbf{f}(\mathbf{x}, \bar{y}) = \sum_{i,y} \alpha_i(y)[\mathbf{f}(\mathbf{x}^{(i)}, y^{(i)})^\top\mathbf{f}(\mathbf{x}, \bar{y}) - \mathbf{f}(\mathbf{x}^{(i)}, y)^\top\mathbf{f}(\mathbf{x}, \bar{y})].$$

For example, we might have basis functions that are polynomial of degree d in terms of image pixels, $f_j(\mathbf{x}, y) = \mathbb{1}(\mathbf{x}_{row_1,col_1} = on \wedge \ldots \wedge \mathbf{x}_{row_d,col_d} = on \wedge y = char)$ for each possible $(row_1, col_1) \ldots (row_d, col_d)$ and $char \in \mathcal{Y}$. Computing this polynomial kernel can be done independently of the dimension d, even though the number of basis functions grows exponentially with d [Vapnik, 1995].

In fact, logistic regression can also be kernelized. However, the hinge loss formulation usually produces sparse solutions in terms of the number of support vectors, while solutions to the corresponding kernelized log-loss problem are generally non-sparse (all examples are support vectors) and require approximations for even relatively small datasets [Wahba *et al.*, 1993; Zhu & Hastie, 2001].

Chapter 3

Structured models

Consider once more the problem of character recognition. In fact, a more natural and useful task is recognizing words and entire sentences. Fig. 3.1 shows an example handwritten word "brace." Distinguishing between the second letter and fourth letter ('r' and 'c') *in isolation* is far from trivial, but in the context of the surrounding letters that together form a word, this task is much less error-prone for humans and should be for computers as well.

In this chapter, we consider prediction problems in which the output is not a single discrete value y, but a set of values $\mathbf{y} = (y_1, \ldots, y_L)$, for example an entire sequence of L characters. For concreteness, let the number of variables L be fixed. The output space $\mathcal{Y} \subseteq \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_L$ we consider is a subset of product of output spaces of single variables. In word recognition, each $\mathcal{Y}_j$ is the alphabet, while $\mathcal{Y}$ is the dictionary. This joint output space is often a proper subset of the product of singleton output spaces, $\mathcal{Y} \subset \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_L$. In word recognition, we might restrict that the letter 'q' never follows by 'z' in English. In addition to "hard" constraints, the output variables are often highly correlated, e.g. consecutive letters in a word. We refer to joint spaces with constraints and correlations as **structured**. We call problems with discrete output spaces **structured classification** or **structured prediction**. **Structured models** we consider in this chapter (and thesis) predict the outputs *jointly*, respecting the constraints and exploiting the correlations in the output space.

The range of prediction problems these broad definitions encompass is immense, arising in fields as diverse as natural language analysis, machine vision, and computational

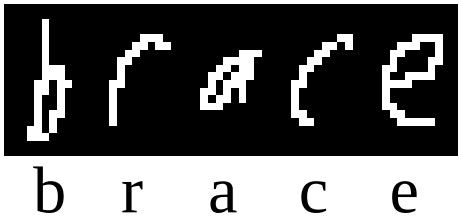


Figure 3.1: Handwritten word recognition: sample from Kassel [1995] data set.

biology, to name a few. The class of structured models $\mathcal{H}$ we consider is essentially of the same form as in previous chapter, except that y has been replaced by $\mathbf{y}$:

$$h_{\mathbf{w}}(\mathbf{x}) = \underset{\mathbf{y}\,:\,\mathbf{g}(\mathbf{x},\mathbf{y})\leq 0}{\arg\max}\ \mathbf{w}^\top \mathbf{f}(\mathbf{x},\mathbf{y}), \tag{3.1}$$

where as before $\mathbf{f}(\mathbf{x},\mathbf{y})$ is a vector of functions $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}^n$. The output space $\mathcal{Y} = \{\mathbf{y} : \mathbf{g}(\mathbf{x},\mathbf{y}) \leq 0\}$ is defined using a vector of functions $\mathbf{g}(\mathbf{x},\mathbf{y})$ that define the constraints, where $\mathbf{g} : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}^k$. This formulation is very general. Clearly, for many $\mathbf{f}, \mathbf{g}$ pairs, finding the optimal $\mathbf{y}$ is intractable. For the most part, we will restrict our attention to models where this optimization problem can be solved in polynomial time. This includes, for example, probabilistic models like Markov networks (in certain cases) and context-free grammars, combinatorial optimization problems like min-cut and matching, convex optimization such as linear, quadratic and semi-definite programming. In other cases, like intractable Markov networks (Ch. 8) and correlation clustering (Ch. 11), we use an *approximate* polynomial time optimization procedure.

3.1 Probabilistic models: generative and conditional

The term **model** is often reserved for probabilistic models, which can be subdivided into generative and conditional with respect to the prediction task. A generative model assigns a normalized joint density $p(\mathbf{x},\mathbf{y})$ to the input and output space $\mathcal{X} \times \mathcal{Y}$ with

$$p(\mathbf{x},\mathbf{y}) \geq 0, \qquad \sum_{\mathbf{y}\in\mathcal{Y}} \int_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x},\mathbf{y}) = 1.$$

A conditional model assigns a normalized density $p(\mathbf{y} \mid \mathbf{x})$ only over the output space $\mathcal{Y}$ with

$$p(\mathbf{y} \mid \mathbf{x}) \geq 0, \qquad \sum_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y} \mid \mathbf{x}) = 1 \qquad \forall \mathbf{x} \in \mathcal{X}.$$

Probabilistic interpretation of the model offers well-understood semantics and an immense toolbox of methods for inference and learning. It also provides an intuitive measure of confidence in the predictions of a model in terms of conditional probabilities. In addition, generative models are typically structured to allow very efficient maximum likelihood learning. A very common class of generative models is the exponential family:

$$p(\mathbf{x}, \mathbf{y}) \propto \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})\}.$$

For exponential families, the maximum likelihood parameters $\mathbf{w}$ with respect to the joint distribution can be computed in closed form using the empirical basis function expectations $\mathbf{E}_S[\mathbf{f}(\mathbf{x}, \mathbf{y})]$ [DeGroot, 1970; Hastie *et al.*, 2001].

Of course, this efficiency comes at a price. Any model is an approximation to the true distribution underlying the data. A generative model must make simplifying assumptions (more precisely, independence assumptions) about the entire $p(\mathbf{x}, \mathbf{y})$, while a conditional model makes many fewer assumption by focusing on $p(\mathbf{y} \mid \mathbf{y})$. Because of this, by optimizing the model to fit the joint distribution $p(\mathbf{x}, \mathbf{y})$, we may be tuning the approximation away from optimal conditional distribution $p(\mathbf{y} \mid \mathbf{x})$, which we use to make the predictions. Given sufficient data, the conditional model will learn the best approximation to $p(\mathbf{y} \mid \mathbf{x})$ possible using $\mathbf{w}$, while the generative model $p(\mathbf{x}, \mathbf{y})$ will not necessarily do so. Typically, however, generative models actually need fewer samples to converge to a good estimate of the joint distribution than conditional models need to accurately represent the conditional distribution. In a regime with very few training samples (relative to the number of parameters $\mathbf{w}$), generative models may actually outperform conditional models [Ng & Jordan, 2001].

3.2 Prediction models: normalized and unnormalized

Probabilistic semantics are certainly not necessary for a good predictive model if we are simply interested in the optimal prediction (the $\arg\max$ in Eq. (3.1)). As we discussed in the previous chapter, support vector machines, which do not represent a conditional distribution, typically perform as well or better than logistic regression [Vapnik, 1995; Cristianini & Shawe-Taylor, 2000].

In general, we can often achieve higher accuracy models when we do not learn a normalized distribution over the outputs, but concentrate on the margin or **decision boundary**, the difference between the optimal $\mathbf{y}$ and the rest. Even more importantly, in many cases we discuss below, normalizing the model (summing over the entire $\mathcal{Y}$) is intractable, while the optimal $\mathbf{y}$ can be found in polynomial time. This fact makes standard maximum likelihood estimation infeasible. The learning methods we advocate in this thesis circumvent this problem by requiring only the maximization problem to be tractable. We still heavily rely on the representation and inference tools familiar from probabilistic models for the construction of and prediction in unnormalized models, but largely dispense with the probabilistic interpretation when needed. Essentially, we use the term **model** very broadly, to include any scheme that assigns scores to the output space $\mathcal{Y}$ and has a procedure for finding the optimal scoring $\mathbf{y}$.

In this chapter, we review basic concepts in probabilistic graphical models called *Markov networks* or *Markov random fields*. We also briefly touch upon examples of context-free grammars and combinatorial problems that will be explained in greater detail in Part III to illustrate the range of prediction problems we address.

3.3 Markov networks

Markov networks provide a framework for a rich family of models for both discrete and continuous prediction [Pearl, 1988; Cowell *et al.*, 1999]. The models treat the inputs and outputs as random variables $\mathbf{X}$ with domain $\mathcal{X}$ and $\mathbf{Y}$ with domain $\mathcal{Y}$ and compactly define a conditional density $p(\mathbf{Y} \mid \mathbf{X})$ or distribution $P(\mathbf{Y} \mid \mathbf{X})$ (we concentrate here on the

conditional Markov networks or CRFs [Lafferty *et al.*, 2001]). The advantage of a *graphical* framework is that it can exploit sparseness in the correlations between outputs Y. The graphical structure of the models encodes the *qualitative* aspects of the distribution: direct dependencies as well as conditional independencies. The *quantitative* aspect of the model is defined by the *potentials* that are associated with nodes and cliques of the graph. Before a formal definition, consider a first-order Markov chain a model for the word recognition task. In Fig. 3.2, the nodes are associated with output variables Y_i and the edges correspond to direct dependencies or correlations. We do not explicitly represent the inputs $\mathbf{X}$ in the figure. For example, the model encodes that Y_j is conditionally independent of the rest of the variables given Y_{j-1}, Y_{j+1}. Intuitively, adjacent letters in a word are highly correlated, but the first-order model is making the assertion (which is certainly an approximation) that once the value of a letter Y_j is known, the correlation between a letter Y_b before j and a letter Y_a after j is negligible. More precisely, we use a model where

$$P(Y_b \mid Y_j, Y_a, \mathbf{x}) = P(Y_b \mid Y_j, \mathbf{x}), \quad P(Y_a \mid Y_j, Y_b, \mathbf{x}) = P(Y_a \mid Y_j, \mathbf{x}), \quad b < j < a.$$

For the purposes of finding the most likely $\mathbf{y}$, this conditional independence property means that the optimization problem is decomposable: given that $Y_j = y_j$, it suffices to *separately* find the optimal subsequence from 1 to j ending with y_j, and the optimal subsequence starting with y_j from j to L.

3.3.1 Representation

The structure of a Markov network is defined by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the nodes are associated with variables $\mathcal{V} = \{Y_1, \ldots, Y_L\}$. A **clique** is a set of nodes $c \subseteq \mathcal{V}$ that form a fully connected subgraph (every two nodes are connected by an edge). Note that each subclique of a clique is also a clique, and we consider each node a singleton clique. In the chain network in Fig. 3.2, the cliques are simply the nodes and the edges: $\mathcal{C}(\mathcal{G}) = \{\{Y_1\}, \ldots, \{Y_5\}, \{Y_1, Y_2\}, \ldots, \{Y_4, Y_5\}\}$. We denote the set of variables in a clique c as $\mathbf{Y}_c$, an assignment of variables in the clique as $\mathbf{y}_c$ and the space of all assignments to the clique as $\mathcal{Y}_c$. We focus on discrete output spaces $\mathcal{Y}$ below, but many of the same representation and inference concepts translate to continuous domains. No assumption is

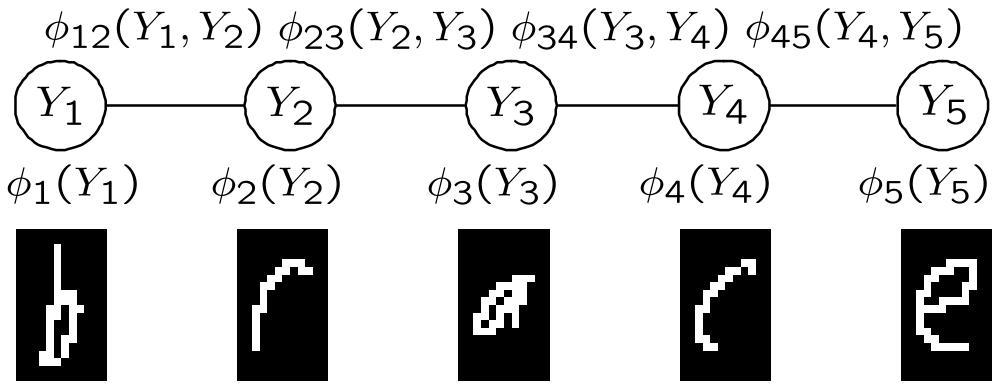


Figure 3.2: First-order Markov chain: $\phi_i(Y_i)$ are node potentials, $\phi_{i,i+1}(Y_i, Y_{i+1})$ are edge potentials (dependence on $\mathbf{x}$ is not shown).

made about $\mathcal{X}$.

Definition 3.3.1 *A **Markov network** is defined by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a set of potentials $\Phi = \{\phi_c\}$. The nodes are associated with variables $\mathcal{V} = \{Y_1, \ldots, Y_L\}$. Each clique $c \in \mathcal{C}(\mathcal{G})$ is associated with a **potential** $\phi_c(\mathbf{x}, \mathbf{y}_c)$ with $\phi_c : \mathcal{X} \times \mathcal{Y}_c \mapsto \mathbb{R}^+$, which specifies a non-negative value for each assignment $\mathbf{y}_c$ to variables in $\mathbf{Y}_c$ and any input $\mathbf{x}$. The Markov network $(\mathcal{G}, \Phi)$ defines a conditional distribution:*

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}(\mathcal{G})} \phi_c(\mathbf{x}, \mathbf{y}_c),$$

*where $\mathcal{C}(\mathcal{G})$ is the set of all the cliques of the graph and $Z(\mathbf{x})$ is the **partition function** given by $Z(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}} \prod_{c \in \mathcal{C}(\mathcal{G})} \phi_c(\mathbf{x}, \mathbf{y}_c)$.*

In our example Fig. 3.2, we have node and edge potentials. Intuitively, the node potentials quantify the correlation between the input $\mathbf{x}$ and the value of the node, while the edge potentials quantify the correlation between the pair of adjacent output variables as well as the input $\mathbf{x}$. Potentials do not have a *local* probabilistic interpretation, but can be thought of as defining an unnormalized score for each assignment in the clique. Conditioned on the image input, appropriate node potentials in our network should give high scores to the correct letters (‘b’,‘r’,‘a’,‘c’,‘e’), though perhaps there would be some ambiguity with the second and fourth letter. For simplicity, assume that the edge potentials would not depend

on the images, but simply should give high scores to pairs of letters that tend to appear often consecutively. Multiplied together, these scores should favor the correct output "brace".

In fact, a Markov network is a generalized log-linear model, since the potentials $\phi_c(\mathbf{x}_c, \mathbf{y}_c)$ could be represented (in log-space) as a sum of basis functions over $\mathbf{x}, \mathbf{y}_c$:

$$\phi_c(\mathbf{x}_c, \mathbf{y}_c) = \exp\left[\sum_{k=1}^{n_c} w_{c,k} f_{c,k}(\mathbf{x}, \mathbf{y}_c)\right] = \exp\left[\mathbf{w}_c^\top \mathbf{f}_c(\mathbf{x}, \mathbf{y}_c)\right]$$

where n_c is the number of basis functions for the clique c. Hence the log of the conditional probability is given by:

$$\log P(\mathbf{y} \mid \mathbf{x}) = \sum_{c \in \mathcal{C}(\mathcal{G})} \mathbf{w}_c^\top \mathbf{f}_c(\mathbf{x}, \mathbf{y}_c) - \log Z_{\mathbf{w}}(\mathbf{x}).$$

In case of node potentials for word recognition, we could use the same basis functions as for individual character recognition: $f_{j,k}(\mathbf{x}, y_j) = \mathbb{1}(\mathbf{x}_{j,row,col} = on \wedge y_j = char)$ for each possible (row, col) in $\mathbf{x}_j$, the window of the image that corresponds to letter j and each $char \in \mathcal{Y}_j$ (we assume the input has been segmented into images $\mathbf{x}_j$ that correspond to letters). In general, we condition a clique only on a portion of the input $\mathbf{x}$, which we denote as $\mathbf{x}_c$. For the edge potentials, we can define basis functions for each combination of letters (assume for simplicity no dependence on $\mathbf{x}$) : $f_{j,j+1,k}(\mathbf{x}, y_j, y_{j+1}) = \mathbb{1}(y_j = char_1 \wedge y_{j+1} = char_2)$ for each $char_1 \in \mathcal{Y}_j$ and $char_2 \in \mathcal{Y}_{j+1}$. In this problem (as well as many others), we are likely to "tie" or "share" the parameters of the model $\mathbf{w}_c$ across cliques. Usually, all single node potentials would share the same weights and basis functions (albeit the relevant portion of the input $\mathbf{x}_c$ is different) and similarly for the pairwise cliques, no matter in what position they appear in the sequence.[1]

With slight abuse of notation, we stack all basis functions into one vector $\mathbf{f}$. For the sequence model, $\mathbf{f}$ has node functions and edge functions, so when c is a node, the edge functions in $\mathbf{f}(\mathbf{x}_c, \mathbf{y}_c)$ are defined to evaluate to zero. Similarly, when c is an edge, the node

[1]Sometimes we might actually want some dependence on the position in the sequence, which can be accomplished by adding more basis functions that condition on the position of the clique.

functions in $\mathbf{f}(\mathbf{x}_c, \mathbf{y}_c)$ are also defined to evaluate to zero. Now we can write:

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{c \in \mathcal{C}(\mathcal{G})} \mathbf{f}(\mathbf{x}_c, \mathbf{y}_c).$$

We stack the weights in the corresponding manner, so the most likely assignment according to the model is given by:

$$\arg\max_{\mathbf{y} \in \mathcal{Y}} \log P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}),$$

in the same form as Eq. (3.1).

3.3.2 Inference

There are several important questions that can be answered by probabilistic models. The task of finding the most likely assignment, known as maximum a-posteriori (MAP) or most likely explanation (MPE), is just one of such questions, but most relevant to our discussion. The **Viterbi** dynamic programming algorithm solves this problem for chain networks in $\mathcal{O}(L)$ time. Let the highest score of any subsequence from 1 to $k > 1$ ending with value y_k be defined as

$$\phi_k^*(y_k) = \max_{\mathbf{y}_{1..k-1}} \prod_j \phi_j(\mathbf{x}, y_j)\phi_j(\mathbf{x}, y_{j-1}, y_j).$$

The algorithm computes the highest scores recursively:

$$\begin{aligned}
&\phi_1^*(y_1) = \phi_1(\mathbf{x}, y_1), \quad \forall y_1 \in \mathcal{Y}_1; \\
&\phi_k^*(y_k) = \max_{y_{k-1} \in \mathcal{Y}_{k-1}} \phi_{k-1}^*(y_{k-1})\phi_j(\mathbf{x}, y_k)\phi_j(\mathbf{x}, y_{k-1}, y_k), \qquad 1 < k \leq L, \ \forall y_k \in \mathcal{Y}_k.
\end{aligned}$$

The highest scoring sequence has score $\max_{y_L} \phi_L^*(y_L)$. Using the $\arg\max$'s of the $\max$'s in the computation of ϕ^*, we can back-trace the highest scoring sequence itself. We assume that score ties are broken in a predetermined way, say according to some lexicographic order of the symbols.

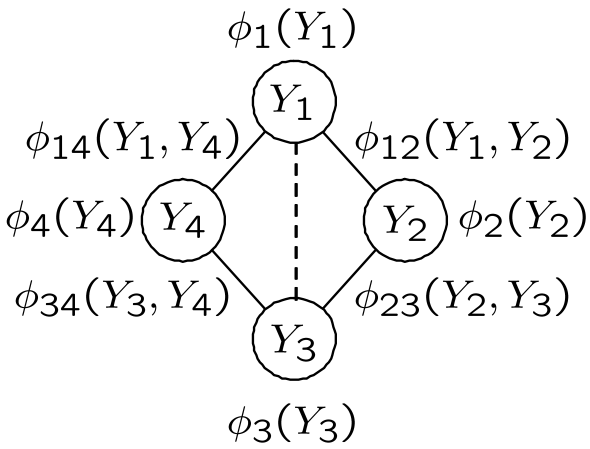


Figure 3.3: Diamond Markov network (added triangulation edge is dashed).

In general Markov networks, MAP inference is NP-hard [Cowell *et al.*, 1999]. However, there are several important subclasses of networks that allow polynomial time inference. The most important of these is the class of networks with *low tree-width*. We need the concept of triangulation (or chordality) to formally define tree-width. Recall that a **cycle** of length l in an undirected graph $\mathcal{G}$ is a sequence of nodes $(v_0, v_1, \ldots, v_l)$, distinct except that $v_0 = v_l$, which are connected by edges $(v_i, v_{i+1}) \in \mathcal{G}$. A *chord* of this cycle is an edge $(v_i, v_j) \in \mathcal{G}$ between non-consecutive nodes.

Definition 3.3.2 (Triangulated graph) *An undirected graph $\mathcal{G}$ is **triangulated** if every one of its cycles of length ≥ 4 possesses a chord.*

Singly-connected graphs, like chains and trees, are triangulated since they contain no cycles. The simplest untriangulated network is the diamond in Fig. 3.3. To triangulate it, we can add the edge (Y_1, Y_3) or (Y_2, Y_4). In general, there are many possible sets of edges that can be added to triangulate a graph. The inference procedure creates a tree of cliques using the graph augmented by triangulation. The critical property of a triangulation for the inference procedure is the size of the largest clique.

Definition 3.3.3 (Tree-width of a graph) *The **tree-width** of a triangulated graph $\mathcal{G}$ is the size of its largest clique minus 1. The **tree-width** of an untriangulated graph $\mathcal{G}$ is the minimum tree-width of all triangulations of $\mathcal{G}$.*

The tree-width of a chain or a tree is 1 and the tree-width of Fig. 3.3 is 2. Finding the minimum tree-width triangulation of a general graph is NP-hard, but good heuristic algorithms exist [Cowell *et al.*, 1999].

The inference procedure is based on a data structure called *junction tree* that can be constructed for a triangulated graph. The junction tree is an alternative representation of the same distribution that allows simple dynamic programming inference similar to the Viterbi algorithm for chains.

Definition 3.3.4 (Junction tree) *A **junction tree** $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ for a triangulated graph $\mathcal{G}$ is a tree in which the nodes are a subset of the cliques of the graph, $\mathcal{V} \subseteq \mathcal{C}(\mathcal{G})$ and the edges $\mathcal{E}$ satisfy the **running intersection property**: for any two cliques c and c', the variables in the intersection $c \cap c'$ are contained in the clique of every node of the tree on the (unique) path between c and c'.*

Fig. 3.4 shows a junction tree for the diamond network. Each of the original clique potentials must associated with exactly one node in the junction tree. For example, the potentials for the $\{Y_1, Y_3, Y_4\}$ and $\{Y_1, Y_3, Y_4\}$ nodes are the product of the associated clique potentials:

$$\begin{aligned}
\phi_{134}(Y_1, Y_3, Y_4) &= \phi_1(Y_1)\phi_4(Y_4)\phi_{14}(Y_1, Y_4)\phi_{34}(Y_3, Y_4), \\
\phi_{123}(Y_1, Y_2, Y_3) &= \phi_2(Y_2)\phi_3(Y_3)\phi_{12}(Y_1, Y_2)\phi_{23}(Y_2, Y_3).
\end{aligned}$$

Algorithms for constructing junction trees from triangulated graphs are described in detail in Cowell *et al.* [1999].

The Viterbi algorithm for junction trees picks an arbitrary root r for the tree $\mathcal{T}$ and proceeds recursively from the leaves to compute the highest scoring subtree at a node by combining the subtrees with highest score from its children. We denote the leaves of the tree as $Lv(\mathcal{T})$ and the children of node c (relative to the root r) as $Ch_r(c)$:

$$\begin{aligned}
&\phi_l^*(\mathbf{y}_l) = \phi_l(\mathbf{x}, \mathbf{y}_l), \quad \forall l \in Lv(\mathcal{T}),\ \forall \mathbf{y}_l \in \mathcal{Y}_l; \\
&\phi_c^*(\mathbf{y}_c) = \phi_c(\mathbf{x}, \mathbf{y}_c) \prod_{c' \in Ch_r(c)} \max_{\mathbf{y}_{c'} \sim \mathbf{y}_c} \phi_{c'}^*(\mathbf{y}_{c'}), \qquad \forall c \in \mathcal{V}(\mathcal{T}) \setminus Lv(\mathcal{T}),\ \forall \mathbf{y}_c \in \mathcal{Y}_c,
\end{aligned}$$

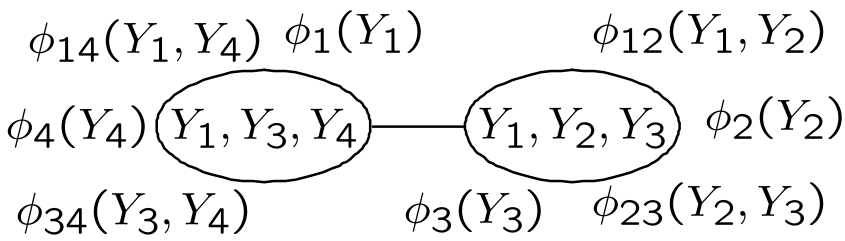


Figure 3.4: Diamond network junction tree. Each of the original potentials is associated with a node in the tree.

where $\mathbf{y}_{c'} \sim \mathbf{y}_c$ denotes whether the partial assignment $\mathbf{y}_c$ is consistent with the partial assignment $\mathbf{y}_{c'}$ on the variables in the intersection of c and c'. The highest score is given by $\max_{\mathbf{y}_r} \phi_r^*(\mathbf{y}_r)$. Using the $\arg\max$'s of the $\max$'s in the computation of ϕ^*, we can back-trace the highest scoring assignment itself. Note that this algorithm is exponential in the tree-width, the size of the largest clique. Similar type of computations using the junction tree can be used to compute the partition function $Z_{\mathbf{w}}(\mathbf{x})$ (by simply replacing $\max$ by $\sum$) as well as marginal probabilities $P(\mathbf{y}_c|\mathbf{x})$ for the cliques of the graph [Cowell *et al.*, 1999].

3.3.3 Linear programming MAP inference

In this section, we present an alternative inference method based on linear programming. Although solving the MAP inference using a general LP solver is less efficient than the dynamic programming algorithms above, this formulation is crucial in viewing Markov networks in a unified framework of the structured models we consider and to our development of common estimation methods in later chapters. Let us begin with a linear integer program to compute the optimal assignment $\mathbf{y}$. We represent an assignment as a set binary variables $\mu_c(\mathbf{y}_c)$, one for each clique c and each value of the clique $\mathbf{y}_c$, that denotes whether the assignment has that value, such that:

$$\log \prod_c \phi_c(\mathbf{x}, \mathbf{y}_c) = \sum_{c, \mathbf{y}_c} \mu_c(\mathbf{y}_c) \log \phi_c(\mathbf{x}, \mathbf{y}_c).$$

$$\mu_2(y_2)$$

0	1	0	0

$$\mu_1(y_1)\left\{\begin{array}{c}0\\0\\1\\0\end{array}\right. \quad \left.\begin{array}{cccc}0&0&0&0\\0&0&0&0\\0&1&0&0\\0&0&0&0\end{array}\right\}\mu_{12}(y_1,y_2)$$

Figure 3.5: Example of marginal agreement: row sums of $\mu_{12}(y_1, y_2)$ agree with $\mu_1(y_1)$, column sums agree with $\mu_2(y_2)$.

We call these variables marginals, as they correspond to the marginals of a distribution that has all of its mass centered on the MAP instantiation (assuming it is unique). There are several elementary constraints that such marginals satisfy. First, they must sum to one for each clique. Second, the marginals for cliques that share variables are consistent. For any clique $c \in \mathcal{C}$ and a subclique $s \subset c$, the assignment of the subclique, $\mu_s(\mathbf{y}_s)$, must be consistent with the assignment of the clique, $\mu_c(\mathbf{y}_c)$. Together, these constraints define a linear integer program:

$$\begin{aligned}
\max \quad & \sum_{c,\mathbf{y}_c} \mu_c(\mathbf{y}_c) \log \phi_c(\mathbf{x}, \mathbf{y}_c) \qquad (3.2)\\
\text{s.t.} \quad & \sum_{\mathbf{y}_c} \mu_c(\mathbf{y}_c) = 1, \quad \forall c \in \mathcal{C}; \qquad \mu_c(\mathbf{y}_c) \in \{0,1\}, \quad \forall c \in \mathcal{C},\ \forall \mathbf{y}_c;\\
& \mu_s(\mathbf{y}_s) = \sum_{\mathbf{y}'_c \sim \mathbf{y}_s} \mu_c(\mathbf{y}'_c), \qquad \forall s, c \in \mathcal{C}, \quad s \subset c, \quad \forall \mathbf{y}_s.
\end{aligned}$$

For example, in case the network is a chain or a tree, we will have node and edge marginals that sum to 1 and agree with each other as in Fig. 3.5.

Clearly, for any assignment $\mathbf{y}'$, we can define $\mu_c(\mathbf{y}_c)$ variables that satisfy the above constraints by setting $\mu_c(\mathbf{y}_c) = \mathbb{1}(\mathbf{y}'_c = \mathbf{y}_c)$. We can also show that converse is true: any valid setting of $\mu_c(\mathbf{y}_c)$ corresponds to a valid assignment $\mathbf{y}$. In fact,

Lemma 3.3.5 *For a triangulated network with unique MAP assignment, the integrality constraint in the integer program in Eq. (3.2) can be relaxed and the resulting LP is guaranteed to have integer solutions.*

A proof of this lemma appears in Wainwright *et al.* [2002]. Intuitively, the constraints force the marginals $\mu_c(\mathbf{y}_c)$ to correspond to some valid joint distribution over the assignments. The optimal distribution with the respect to the objective puts all its mass on the MAP assignment. If the MAP assignment is not unique, the value of the LP is the same as the value of the integer program, and any linear combination of the MAP assignments maximizes the LP.

In case the network is not triangulated, the set of marginals is not guaranteed to represent a valid distribution. Consider, for example, the diamond network in Fig. 3.3 with binary variables, with the following edge marginals that are consistent with the constraints:

$$\begin{aligned}
\mu_{12}(0,0) &= \mu_{12}(1,1) = 0.5, & \mu_{12}(1,0) &= \mu_{12}(0,1) = 0;\\
\mu_{23}(0,0) &= \mu_{23}(1,1) = 0.5, & \mu_{23}(1,0) &= \mu_{23}(0,1) = 0;\\
\mu_{34}(0,0) &= \mu_{34}(1,1) = 0.5, & \mu_{34}(1,0) &= \mu_{34}(0,1) = 0;\\
\mu_{14}(0,0) &= \mu_{34}(1,1) = 0, & \mu_{14}(1,0) &= \mu_{14}(0,1) = 0.5.
\end{aligned}$$

The corresponding node marginals must all be set to 0.5. Note that the edge marginals for $(1,2),(2,3),(3,4)$ disallow any assignment other than 0000 or 1111, but the edge marginal for $(1,4)$ disallows any assignment that has $Y_1 = Y_4$. Hence this set of marginals disallows all assignments. If we triangulate the graph and add the cliques $\{Y_1, Y_2, Y_3\}$ and $\{Y_1, Y_3, Y_4\}$ with their corresponding constraints, the above marginals will be disallowed.

In graphs where triangulation produces very large cliques, exact inference is intractable. We can resort to the above LP *without* triangulation as an approximate inference procedure (augmented with some procedure for rounding possibly fractional solutions). In Ch. 7, we discuss another subclass of networks where MAP inference using LPs is tractable for any network topology, but with a restricted type of potentials.

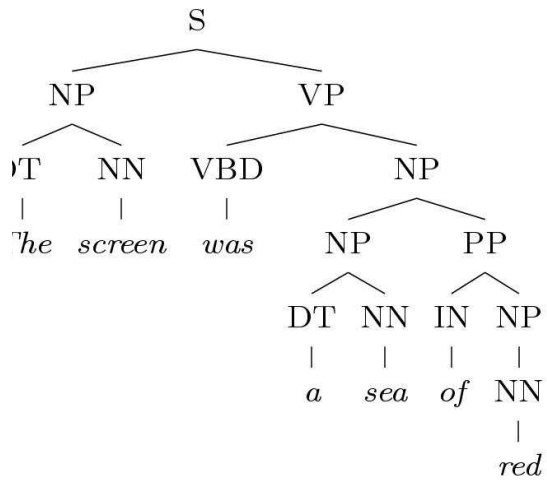


Figure 3.6: Example parse tree from Penn Treebank [Marcus *et al.*, 1993].

3.4 Context free grammars

Context-free grammars are one of the primary formalisms for capturing the recursive structure of syntactic constructions [Manning & Schütze, 1999]. For example, Fig. 3.6 shows a parse tree for the sentence *The screen was a sea of red*. This tree is from the Penn Treebank [Marcus *et al.*, 1993], a primary linguistic resource for expert-annotated English text. The non-terminal symbols (labels of internal nodes) correspond to syntactic categories such as noun phrase (NP), verbal phrase (VP) or prepositional phrase (PP) and part-of-speech tags like nouns (NN), verbs (VBD), determiners (DT) and prepositions (IN). The terminal symbols (leaves) are the words of the sentence.

For clarity of presentation, we restrict our grammars to be in Chomsky normal form[2](CNF), where all rules in the grammar are of the form: $A \rightarrow B\ C$ and $A \rightarrow D$, where A, B and C are non-terminal symbols, and D is a terminal symbol.

Definition 3.4.1 (CFG) *A CFG $\mathcal{G}$ consists of:*

- *A set of non-terminal symbols, $\mathcal{N}$*
- *A designated set of start symbols, $\mathcal{N}_S \subseteq \mathcal{N}$*

[2]Any CFG can be represented by another CFG in CNF that generates the same set of sentences.

- *A set of terminal symbols,* $\mathcal{T}$
- *A set of productions,* $\mathcal{P} = \{\mathcal{P}_B, \mathcal{P}_U\}$*, divided into*
  - *Binary productions,* $\mathcal{P}_B = \{A \rightarrow B\ C : A, B, C \in \mathcal{N}\}$ *and*
  - *Unary productions,* $\mathcal{P}_U = \{A \rightarrow D : A \in \mathcal{N}, D \in \mathcal{T}\}$.

Consider a very simple grammar:

- $\mathcal{N} = \{\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \mathrm{PP}, \mathrm{NN}, \mathrm{VBD}, \mathrm{DT}, \mathrm{IN}\}$
- $\mathcal{N}_S = \{\mathrm{S}\}$
- $\mathcal{T} = \{\mathrm{The}, \mathrm{the}, \mathrm{cat}, \mathrm{dog}, \mathrm{tree}, \mathrm{saw}, \mathrm{from}\}$
- $\mathcal{P}_B = \{\mathrm{S} \rightarrow \mathrm{NP\ VP}, \mathrm{NP} \rightarrow \mathrm{DT\ NN}, \mathrm{NP} \rightarrow \mathrm{NP\ PP}, \mathrm{VP} \rightarrow \mathrm{VBD\ NP}, \mathrm{VP} \rightarrow \mathrm{VP\ PP}, \mathrm{PP} \rightarrow \mathrm{IN\ NP}\}$.
- $\mathcal{P}_U = \{\mathrm{DT} \rightarrow \mathrm{The}, \mathrm{DT} \rightarrow \mathrm{the}, \mathrm{NN} \rightarrow \mathrm{cat}, \mathrm{NN} \rightarrow \mathrm{dog}, \mathrm{NN} \rightarrow \mathrm{tree}, \mathrm{VBD} \rightarrow \mathrm{saw}, \mathrm{IN} \rightarrow \mathrm{from}\}$

A grammar generates a sentence by starting with a symbol in $\mathcal{N}_S$ and applying the productions in $\mathcal{P}$ to rewrite nonterminal symbols. For example, we can generate *The cat saw the dog* by starting with $\mathrm{S} \rightarrow \mathrm{NP\ VP}$, rewriting the NP as $\mathrm{NP} \rightarrow \mathrm{DT\ NN}$ with $\mathrm{DT} \rightarrow \mathrm{The}$ and $\mathrm{NN} \rightarrow \mathrm{cat}$, then rewriting the VP as $\mathrm{VP} \rightarrow \mathrm{VBD\ NP}$ with $\mathrm{VBD} \rightarrow \mathrm{saw}$, again using $\mathrm{NP} \rightarrow \mathrm{DT\ NN}$, but now with $\mathrm{DT} \rightarrow \mathrm{the}$ and $\mathrm{NN} \rightarrow \mathrm{dog}$. We can represent such derivations using trees like in Fig. 3.6 or (more compactly) using bracketed expressions like the one below:

$$[[\mathrm{The}_{\mathrm{DT}}\ \mathrm{cat}_{\mathrm{NN}}]_{\mathrm{NP}}\ [\mathrm{saw}_{\mathrm{VBD}}\ [\mathrm{the}_{\mathrm{DT}}\ \mathrm{dog}_{\mathrm{NN}}]_{\mathrm{NP}}]_{\mathrm{VP}}]_{\mathrm{S}}\,.$$

The simple grammar above can generate sentences of arbitrary length, since it has several recursive productions. It can also generate the same sentence several ways. In general, there are exponentially many parse trees that produce a sentence of length l. Consider the sentence *The cat saw the dog from the tree*. The likely analysis of the sentence is that the cat, sitting in the tree, saw the dog. An unlikely but possible alternative is that the cat

actually saw the dog who lived near the tree or was tied to it in the past. Our grammar allows both interpretations, with the difference being in the analysis of the top-level VP:

$$[\text{saw}_{\text{VBD}}\ [\text{the}_{\text{DT}}\ \text{dog}_{\text{NN}}]_{\text{NP}}]_{\text{VP}}\ \ [[\text{from}_{\text{IN}}\ [\text{the}_{\text{DT}}\ \text{tree}_{\text{NN}}]_{\text{NP}}]_{\text{PP}},$$
$$\text{saw}_{\text{VBD}}\ [[\text{the}_{\text{DT}}\ \text{dog}_{\text{NN}}]_{\text{NP}}\ [\text{from}_{\text{IN}}\ [\text{the}_{\text{DT}}\ \text{tree}_{\text{NN}}]_{\text{NP}}]_{\text{PP}}]_{\text{NP}}.$$

This kind of ambiguity, called prepositional attachment, is very common in many realistic grammars. A standard approach to resolving ambiguity is to use a PCFG to define a joint probability distribution over the space of parse trees $\mathcal{Y}$ and sentences $\mathcal{X}$. Standard PCFG parsers use a Viterbi-style algorithm to compute $\arg\max_{\mathbf{y}} P(\mathbf{x}, \mathbf{y})$ as the most likely parse tree for a sentence $\mathbf{x}$. The distribution $P(\mathbf{x}, \mathbf{y})$ is defined by assigning a probability to each production and making sure that the sum of probabilities of all productions starting with a each symbol is 1:

$$\sum_{B,C:A\to B\ C\in\mathcal{P}_B} P(A \to B\ C) = 1, \qquad \sum_{D:A\to D\in\mathcal{P}_U} P(A \to D\ ) = 1, \qquad \forall A \in \mathcal{N}.$$

We also need to assign a probability to the different starting symbols $P(A) \in \mathcal{N}_S$ such that $\sum_{A\in\mathcal{N}_S} P(A) = 1$. The probability of a tree is simply the product of probabilities of the productions used in the tree (times the probability of the starting symbol). Hence the log-probability of a tree is a sum of the log-probabilities of its productions. By letting our basis functions $\mathbf{f}(\mathbf{x}, \mathbf{y})$ consist of the counts of the productions and $\mathbf{w}$ be their log-probabilities, we can cast PCFG as a structured linear model (in log space). In Ch. 9, we will show how to represent a parse tree as an assignment of variables $\mathbf{Y}$ with appropriate constraints to express PCFGS (and more generally weighted CFGs) in the form of Eq. (3.1) as

$$h_{\mathbf{w}}(\mathbf{x}) = \underset{\mathbf{y}\,:\,\mathbf{g}(\mathbf{x},\mathbf{y})\leq 0}{\arg\max}\ \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}),$$

and describe the associated algorithm to compute the highest scoring parse tree $\mathbf{y}$ given a sentence $\mathbf{x}$.

3.5 Combinatorial problems

Many important computational tasks are formulated as combinatorial optimization problems such as the maximum weight bipartite and perfect matching, spanning tree, graph-cut, edge-cover, bin-packing, and many others [Lawler, 1976; Papadimitriou & Steiglitz, 1982; Cormen *et al.*, 2001]. These problems arise in applications such as resource allocation, job assignment, routing, scheduling, network design and many more. In some domains, the weights of the objective function in the optimization problem are simple and natural to define (for example, Euclidian distance or temporal latency), but in many others, constructing the weights is an important and labor-intensive design task. Treated abstractly, a combinatorial space of structures, such as matchings or graph-cuts or trees), together with a scoring scheme that assigns weights to candidate outputs is a kind of a model.

As a particularly simple and relevant example, consider modeling the task of assigning reviewers to papers as a maximum weight bipartite matching problem, where the weights represent the "expertise" of each reviewer for each paper. More specifically, suppose we would like to have R reviewers per paper, and that each reviewer be assigned at most P papers. For each paper and reviewer, we have an a weight q_{jk} indicating the qualification level of reviewer j for evaluating paper k. Our objective is to find an assignment for reviewers to papers that maximizes the total weight. We represent a matching with a set of binary variables y_{jk} that take the value 1 if reviewer j is assigned to paper k, and 0 otherwise. The bipartite matching problem can be solved using a combinatorial algorithm or the following linear program:

$$\begin{aligned} \max \quad & \sum_{j,k} \mu_{jk} q_{jk} \\ \text{s.t.} \quad & \sum_{j} \mu_{jk} = R, \qquad \sum_{k} \mu_{jk} \leq P, \qquad 0 \leq \mu_{jk} \leq 1. \end{aligned} \tag{3.3}$$

This LP is guaranteed to produce integer solutions (as long as P and R are integers) for any weights $q(\mathbf{y})$ [Nemhauser & Wolsey, 1999].

The quality of the solution found depends critically on the choice of weights that define the objective. A simple scheme could measure the "expertise" as the percent of word

overlap in the reviewer's home page and the paper's abstract. However, we would want to weight certain words much more (words that are relevant to the subject and infrequent). Constructing and tuning the weights for a problem is a difficult and time-consuming process, just as it is for Markov networks for handwriting recognition.

As usual, we will represent the objective $q(\mathbf{y})$ as a weighted combination of a set of basis functions $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$. Let $\mathbf{x}_{jk}$ denote the intersection of the set of words occurring in $webpage(j) \cap abstract(k)$, the web page of a reviewer j and the abstract of the paper k. We can define $f_d(\mathbf{x}, \mathbf{y}) = \sum_{jk} y_{jk} \mathbb{1}(word_d \in \mathbf{x}_{jk})$, the number of times word d was in both the web page of a reviewer and the abstract of the paper that were matched in $\mathbf{y}$. Then the score q_{jk} is simply $q_{jk} = \sum_d w_d \mathbb{1}(word_d \in \mathbf{x}_{jk})$, a weighted combination of overlapping words. In the next chapter we will show how to learn the parameters $\mathbf{w}$ in much the same way we learn the parameters $\mathbf{w}$ of a Markov network.

The space of bipartite matchings illustrates an important property of many structured spaces: the maximization problem $\arg\max_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$ is easier than the normalization problem $\sum_{\mathbf{y} \in \mathcal{Y}} \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})\}$. The maximum weight bipartite matching can be found in polynomial (cubic) time in the number of nodes in the graph using a combinatorial algorithm. However, even simply counting the number of matchings is #P-complete [Valiant, 1979; Garey & Johnson, 1979]. Note that counting is easier than normalization, which is essentially weighted counting. This fact makes a probabilistic interpretation of the model as a distribution over matchings intractable to compute. Similarly, exact maximum likelihood estimation is intractable, since it requires computing the normalization.

Chapter 4

Structured maximum margin estimation

In the previous chapter, we described several important types of structured models of the form:

$$h_{\mathbf{w}}(\mathbf{x}) = \underset{\mathbf{y}\,:\,\mathbf{g}(\mathbf{x},\mathbf{y})\leq 0}{\arg\max}\ \mathbf{w}^\top \mathbf{f}(\mathbf{x},\mathbf{y}), \tag{4.1}$$

where we assume that the optimization problem $\max_{\mathbf{y}\,:\,\mathbf{g}(\mathbf{x},\mathbf{y})\leq 0} \mathbf{w}^\top \mathbf{f}(\mathbf{x},\mathbf{y})$ can be solved or approximated by a compact convex optimization problem for some convex subset of parameters $\mathbf{w} \in \mathcal{W}$. A *compact* problem formulation is polynomial in the description length of the objective and the constraints.

Given a sample $S = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^m$, we develop methods for finding parameters $\mathbf{w}$ such that:

$$\underset{\mathbf{y}\in\mathcal{Y}^{(i)}}{\arg\max}\, \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y}) \ \approx\ \mathbf{y}^{(i)}, \quad \forall i,$$

where $\mathcal{Y}^{(i)} = \{\mathbf{y} : \mathbf{g}(\mathbf{x}^{(i)}, \mathbf{y}) \leq 0\}$. In this chapter, we describe at an abstract level two general approaches to structured estimation that we apply in the rest of the thesis. Both of these approaches define a convex optimization problem for finding such parameters $\mathbf{w}$.

There are several reasons to derive compact convex formulations. First and foremost, we can find globally optimal parameters (with fixed precision) in polynomial time. Second, we can use standard optimization software to solve the problem. Although special-purpose algorithms that exploit the structure of a particular problem are often much faster

(see Ch. 6), the availability of off-the-shelf software is very important for quick development and testing of such models. Third, we can analyze the generalization performance of the framework without worrying about the actual algorithms used to carry out the optimization and the associated woes of intractable optimization problems: local minima, greedy and heuristic methods, etc.

Our framework applies not only to the standard models typically estimated by probabilistic methods, such as Markov networks and context-free grammars, but also to a wide range of "unconventional" predictive models. Such models include graph cuts and weighted matchings, where maximum likelihood estimation is intractable. We provide exact maximum margin solutions for several of these problems (Ch. 7 and Ch. 10).

In prediction problems where the maximization in Eq. (4.1) is intractable, we consider convex programs that provide only an upper or lower bound on the true solution. We discuss how to use these approximate solutions for approximate learning of parameters.

4.1 Max-margin estimation

As in the univariate prediction, we measure the error of approximation using a loss function ℓ. In structured problems, where we are jointly predicting multiple variables, the loss is often not just the simple 0-1 loss or squared error. For structured classification, a natural loss function is a kind of Hamming distance between $\mathbf{y}^{(i)}$ and $h(\mathbf{x}^{(i)})$: the number of variables predicted incorrectly. We will explore these and more general loss functions in the following chapters.

4.1.1 Min-max formulation

Throughout, we will adopt the hinge upper bound $\overline{\ell}_i(h(\mathbf{x}^{(i)}))$ on the loss function for structured classification inspired by max-margin criterion:

$$\overline{\ell}_i(h(\mathbf{x}^{(i)})) = \max_{\mathbf{y}\in\mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})] - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) \geq \ell_i(h(\mathbf{x}^{(i)})),$$

where as before, $\overline{\ell}_i(h(\mathbf{x}^{(i)})) = \overline{\ell}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, h(\mathbf{x}^{(i)}))$, $\ell_i(h(\mathbf{x}^{(i)})) = \ell(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, h(\mathbf{x}^{(i)}))$, and $\mathbf{f}_i(\mathbf{y}) = \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y})$. With this upper bound, the min-max formulation for structured classification problem is analogous to multi-class SVM formulation in Eq. (2.9) and Eq. (2.10):

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq \max_{\mathbf{y}\in\mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})], \quad \forall i. \end{aligned} \tag{4.2}$$

The above formulation is a convex quadratic program in $\mathbf{w}$, since $\max_{\mathbf{y}\in\mathcal{Y}^{(i)}}[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$ is convex in $\mathbf{w}$ (maximum of affine functions is a convex function). For brevity, we did not explicitly include the constraint that the parameters are in some legal convex set ($\mathbf{w} \in \mathcal{W}$, most often $\mathbb{R}^n$), but assume this throughout this chapter.

The problem with Eq. (4.2) is that the constraints have a very unwieldy form. Another way to express this problem is using $\sum_i |\mathcal{Y}^{(i)}|$ linear constraints, which is generally exponential in L_i, the number of variables in $\mathbf{y}_i$.

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y}), \quad \forall i, \ \ \forall \mathbf{y} \in \mathcal{Y}^{(i)}. \end{aligned} \tag{4.3}$$

This form reveals the "maximum margin" nature of the formulation. We can interpret $\frac{1}{||\mathbf{w}||}\mathbf{w}^\top[\mathbf{f}_i(\mathbf{y}^{(i)}) - \mathbf{f}_i(\mathbf{y})]$ as the *margin* of $\mathbf{y}^{(i)}$ over another $\mathbf{y} \in \mathcal{Y}^{(i)}$. Assuming ξ_i are all zero (say because C is very large), the constraints enforce

$$\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) \geq \ell_i(\mathbf{y}),$$

so minimizing $||\mathbf{w}||$ maximizes the smallest such margin, scaled by the loss $\ell_i(\mathbf{y})$. The slack variables ξ_i allow for violations of the constraints at a cost $C\xi_i$. If the loss function is not uniform over all the mistakes $\mathbf{y} \neq \mathbf{y}^{(i)}$, then the constraints make costly mistakes (those with high $\ell_i(\mathbf{y})$) less likely. In Ch. 5 we analyze the effect of non-uniform loss function (Hamming distance type loss) on generalization, and show a strong connection between the loss-scaled margin and expected risk of the learned model.

The formulation in Eq. (4.3) is a standard QP with linear constraints, but its exponential

size is in general prohibitive. We now return to Eq. (4.2) and transform it to a a more manageable problem. The key to solving Eq. (4.2) efficiently is the **loss-augmented** inference

$$\max_{\mathbf{y}\in\mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]. \tag{4.4}$$

Even if $\max_{\mathbf{y}\in\mathcal{Y}^{(i)}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$ can be solved in polynomial time using convex optimization, the form of the loss term $\ell_i(\mathbf{y})$ is crucial for the loss-augmented inference to remain tractable. The range of tractable losses will depend strongly on the problem itself ($\mathbf{f}$ and $\mathcal{Y}$). Even within the range of tractable losses, some are more efficiently computable than others. A large part of the development of structured estimation methods in the following chapters is identifying appropriate loss functions for the application and designing convex formulations for the loss-augmented inference.

Assume that we find such a formulation in terms of a set of variables μ_i, with a concave (in μ_i) objective $\widetilde{f}_i(\mathbf{w}, \mu_i)$ and subject to convex constraints $\widetilde{\mathbf{g}}_i(\mu_i)$:

$$\max_{\mathbf{y}\in\mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})] \quad = \quad \max_{\mu_i:\widetilde{\mathbf{g}}_i(\mu_i)\leq 0} \quad \widetilde{f}_i(\mathbf{w}, \mu_i). \tag{4.5}$$

We call such formulation compact if the number of variables μ_i and constraints $\widetilde{\mathbf{g}}_i(\mu_i)$ is polynomial in L_i, the number of variables in $\mathbf{y}^{(i)}$.

Note that $\max_{\mu_i:\widetilde{\mathbf{g}}_i(\mu_i)\leq 0} \widetilde{f}_i(\mathbf{w}, \mu_i)$ must be convex in $\mathbf{w}$, since Eq. (4.4) is. Likewise, we can assume that it is feasible and bounded if Eq. (4.4) is. In the next section, we develop a max-margin formulation that uses Lagrangian duality (see [Boyd & Vandenberghe, 2004] for an excellent review) to define a joint, compact convex problem for estimating the parameters $\mathbf{w}$.

To make the symbols concrete, consider the example of the reviewer-assignment problem we discussed in the previous chapter: we would like a bipartite matching with R reviewers per paper and at most P papers per reviewer. Each training sample i consists of a matching of $N_p^{(i)}$ papers and $N_r^{(i)}$ reviewers from some previous year. Let $\mathbf{x}_{jk}$ denote the intersection of the set of words occurring in the web page of a reviewer j and the abstract of the paper k. Let y_{jk} indicate whether reviewer j is matched to the paper k. We can define a basis function $f_d(\mathbf{x}_{jk}, y_{jk}) = y_{jk}\mathbb{1}(word_d \in \mathbf{x}_{jk})$, which indicates whether the word d is

in both the web page of a reviewer and the abstract of the paper that are matched in $\mathbf{y}$. We abbreviate the vector of all the basis functions for each edge jk as $y_{jk}\mathbf{f}_{jk}^{(i)} = \mathbf{f}(\mathbf{x}_{jk}^{(i)}, y_{jk})$.

We assume that the loss function decomposes over the variables y_{ij}. For example, the Hamming loss simply counts the number of different edges in the matchings $\mathbf{y}$ and $\mathbf{y}^{(i)}$:

$$\ell_i^H(\mathbf{y}) = \sum_{jk} \ell_{i,jk}^{0/1}(y_{jk}) = \sum_{jk} \mathbb{1}(y_{jk} \neq y_{jk}^{(i)}) = RN_p^{(i)} - \sum_{jk} y_{jk} y_{jk}^{(i)}.$$

The last equality follows from the fact that any valid matching for example i has R reviewers for $N_p^{(i)}$ papers, hence $RN_p^{(i)} - \sum_{jk} y_{jk} y_{jk}^{(i)}$ represents exactly the number of edges that are different between $\mathbf{y}$ and $\mathbf{y}^{(i)}$. Combining the two pieces, we have

$$\mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y}) = \sum_{jk} [\mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}, y_{jk}) + \ell_{i,jk}^{0/1}(y_{jk})] = RN_p^{(i)} + \sum_{jk} y_{jk} [\mathbf{w}^\top \mathbf{f}_{jk} - y_{jk}^{(i)}].$$

The loss-augmented inference problem can be then written as an LP in μ_i similar to Eq. (3.3) (without the constant term $RN_p^{(i)}$):

$$\begin{aligned} \max \quad & \sum_{jk} \mu_{i,jk} [\mathbf{w}^\top \mathbf{f}_{jk} - y_{jk}^{(i)}] \\ \text{s.t.} \quad & \sum_{j} \mu_{i,jk} = R, \qquad \sum_{k} \mu_{i,jk} \leq P, \qquad 0 \leq \mu_{i,jk} \leq 1. \end{aligned}$$

In terms of Eq. (4.5), $\widetilde{f}_i$ and $\widetilde{\mathbf{g}}_i$ are affine in μ_i: $\widetilde{f}_i(\mathbf{w}, \mu_i) = RN_p^{(i)} + \sum_{i,j} \mu_{i,jk} [\mathbf{w}^\top \mathbf{f}_{jk} - y_{jk}^{(i)}]$ and $\widetilde{\mathbf{g}}_i(\mu_i) \leq 0 \Leftrightarrow \sum_j \mu_{i,jk} = R,\ \sum_k \mu_{i,jk} \leq P,\ 0 \leq \mu_{i,jk} \leq 1$.

In general, when we can express $\max_{\mathbf{y} \in \mathcal{Y}^{(i)}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y})$ as an LP and we use a loss function this is linear in the number of mistakes, we have a linear program of this form for the loss-augmented inference:

$$d_i + \max\ (\mathbf{F}_i \mathbf{w} + \mathbf{c}_i)^\top \mu_i \quad \text{s.t.} \quad \mathbf{A}_i \mu_i \leq \mathbf{b}_i,\ \mu_i \geq 0, \tag{4.6}$$

for appropriately defined $d_i, \mathbf{F}_i, \mathbf{c}_i, \mathbf{A}_i, \mathbf{b}_i$, which depend only on $\mathbf{x}^{(i)}$, $\mathbf{y}^{(i)}$, $\mathbf{f}(\mathbf{x}, \mathbf{y})$ and $\mathbf{g}(\mathbf{x}, \mathbf{y})$. Note that the dependence on $\mathbf{w}$ is linear and only in the objective of the LP. If this LP is compact (the number of variables and constraints is polynomial in the number of

label variables), then we can use it to solve the max-margin estimation problem efficiently by using convex duality.

The **Lagrangian** associated with Eq. (4.4) is given by

$$L_{i,\mathbf{w}}(\mu_i, \lambda_i) = \widetilde{f}_i(\mathbf{w}, \mu_i) - \lambda_i^\top \widetilde{\mathbf{g}}_i(\mu_i), \tag{4.7}$$

where $\lambda_i \geq 0$ is a vector of **Lagrange multipliers**, one for each constraint function in $\widetilde{\mathbf{g}}_i(\mu_i)$. Since we assume that $\widetilde{f}_i(\mathbf{w}, \mu_i)$ is concave in μ_i and bounded on the non-empty set $\mu_i : \widetilde{\mathbf{g}}_i(\mu_i) \leq 0$, we have **strong duality**:

$$\max_{\mu_i : \widetilde{\mathbf{g}}_i(\mu_i) \leq 0} \widetilde{f}_i(\mathbf{w}, \mu_i) = \min_{\lambda_i \geq 0} \max_{\mu_i} L_{i,\mathbf{w}}(\mu_i, \lambda_i).$$

For many forms of $\widetilde{f}$ and $\widetilde{\mathbf{g}}$, we can write the Lagrangian dual $\min_{\lambda_i \geq 0} \max_{\mu_i} L_{i,\mathbf{w}}(\mu_i, \lambda_i)$ explicitly as:

$$\begin{aligned} \min \quad & h_i(\mathbf{w}, \lambda_i) \\ \text{s.t.} \quad & \mathbf{q}_i(\mathbf{w}, \lambda_i) \leq 0, \end{aligned} \tag{4.8}$$

where $h_i(\mathbf{w}, \lambda_i)$ and $\mathbf{q}_i(\mathbf{w}, \lambda_i)$ are convex in both $\mathbf{w}$ and λ_i. (We folded $\lambda_i \geq 0$ into $\mathbf{q}_i(\mathbf{w}, \lambda_i)$ for brevity.) Since the original problem had polynomial size, the dual is polynomial size as well. For example, the dual of the LP in Eq. (4.6) is

$$d_i + \min \mathbf{b}_i^\top \lambda_i \quad \text{s.t.} \quad \mathbf{A}_i^\top \lambda_i \geq \mathbf{F}_i \mathbf{w} + \mathbf{c}_i, \ \lambda_i \geq 0, \tag{4.9}$$

where $h_i(\mathbf{w}, \lambda_i) = d_i + \mathbf{b}_i^\top \lambda_i$ and $\mathbf{q}_i(\mathbf{w}, \lambda_i) \leq 0$ is $\{\mathbf{F}_i \mathbf{w} + \mathbf{c}_i - \mathbf{A}_i^\top \lambda_i \leq 0, -\lambda_i \leq 0\}$.

Plugging Eq. (4.8) into Eq. (4.2), we get

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_i \xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq \min_{\mathbf{q}_i(\mathbf{w}, \lambda_i) \leq 0} h_i(\mathbf{w}, \lambda_i), \quad \forall i. \end{aligned} \tag{4.10}$$

Moreover, we can combine the minimization over λ with minimization over $\{\mathbf{w}, \xi\}$. The

reason for this is that if the right hand side is not at the minimum, the constraint is tighter than necessary, leading to a suboptimal solution $\mathbf{w}$. Optimizing jointly over λ as well will produce a solution to $\{\mathbf{w}, \xi\}$ that is optimal.

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq h_i(\mathbf{w}, \lambda_i), \quad \forall i; \\ & \mathbf{q}_i(\mathbf{w}, \lambda_i) \leq 0, \quad \forall i. \end{aligned} \tag{4.11}$$

Hence we have a joint and compact convex optimization program for estimating $\mathbf{w}$. The exact form of this program depends strongly on $\widetilde{f}$ and $\widetilde{\mathbf{g}}$. For our LP-based example, we have a QP with linear constraints:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq d_i + \mathbf{b}_i^\top \lambda_i, \quad \forall i; \\ & \mathbf{A}_i^\top \lambda_i \geq \mathbf{F}_i \mathbf{w} + \mathbf{c}_i, \quad \forall i; \\ & \lambda_i \geq 0, \quad \forall i. \end{aligned} \tag{4.12}$$

4.1.2 Certificate formulation

In the previous section, we assumed a *compact* convex formulation of the loss-augmented max in Eq. (4.4). There are several important combinatorial problems which allow polynomial time solution yet do not have a compact convex optimization formulation. For example, maximum weight perfect (non-bipartite) matching and spanning tree problems can be expressed as linear programs with *exponentially* many constraints, but no polynomial formulation is known [Bertsimas & Tsitsiklis, 1997; Schrijver, 2003]. Both of these problems, however, can be solved in polynomial time using combinatorial algorithms. In some cases, though, we can find a compact *certificate of optimality* that guarantees that $\mathbf{y}^{(i)} = \arg\max_{\mathbf{y}}[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$ without expressing loss-augmented inference as a compact convex program. Intuitively, just verifying that a given assignment is optimal is sometimes easier than actually finding it.

Consider the maximum weight spanning tree problem. A basic property of a spanning tree is that cutting any edge (j,k) in the tree creates two disconnected sets of nodes $(\mathcal{V}_j[jk], \mathcal{V}_k[jk])$, where $j \in \mathcal{V}_j[jk]$ and $k \in \mathcal{V}_k[jk]$. A spanning tree is optimal with respect to a set of edge weights if and only if for every edge (j,k) in the tree connecting $\mathcal{V}_j[jk]$ and $\mathcal{V}_k[jk]$, the weight of (j,k) is larger than (or equal to) the weight of any other edge (j',k') in the graph with $j' \in \mathcal{V}_j[jk], k' \in \mathcal{V}_k[jk]$ [Cormen *et al.*, 2001]. We discuss the conditions for optimality of perfect matchings in Ch. 10. Suppose that we can find a *compact* convex formulation of these conditions via a polynomial (in L_i) set of functions $\mathbf{q}_i(\mathbf{w}, \nu_i)$, jointly convex in $\mathbf{w}$ and auxiliary variables ν_i:

$$\exists \nu_i \ \text{ s.t.} \quad \mathbf{q}_i(\mathbf{w}, \nu_i) \leq 0 \quad \Leftrightarrow \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y}), \quad \forall \mathbf{y} \in \mathcal{Y}^{(i)}.$$

Then the following joint convex program in $\mathbf{w}$ and ν computes the max-margin parameters:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{s.t.} \quad & \mathbf{q}_i(\mathbf{w}, \nu_i) \leq 0, \quad \forall i. \end{aligned} \tag{4.13}$$

Expressing the spanning tree optimality does not require additional variables ν_i, but in other problems, such as in perfect matching optimality in Ch. 10, such auxiliary variables are needed. In the spanning tree problem, suppose y_{jk} encodes whether edge (j,k) is in the tree and the score of the edge is given by $\mathbf{w}^\top \mathbf{f}_{i,jk}$ for some basis functions $\mathbf{f}(\mathbf{x}^{(i)}_{jk}, y_{jk})$. We also assume that the loss function decomposes into a sum of losses over the edges, with loss for each wrong edge given by $\ell_{i,jk}$. Then the optimality conditions are:

$$\mathbf{w}^\top \mathbf{f}_{i,jk} \geq \mathbf{w}^\top \mathbf{f}_{i,j'k'} + \ell_{i,j'k'}, \quad \forall jk, j'k' \text{ s.t. } y^{(i)}_{jk} = 1, \ j' \in \mathcal{V}_j[jk], \ k' \in \mathcal{V}_k[jk].$$

For a full graph, we have $\left(|\mathcal{V}^{(i)}|^3\right)$ constraints for each example i, where $|\mathcal{V}^{(i)}|$ is the number of nodes in the graph for example i.

Note that this formulation does not allow for violations of the margin constraints (it has no slack variables ξ_i). If the basis functions are not sufficiently rich to ensure that each $\mathbf{y}^{(i)}$ is optimal, then Eq. (4.1.2) may be infeasible. Essentially, this formulation requires that the upper bound on the empirical risk be zero, $\mathcal{R}^{\bar{\ell}}_S[h_{\mathbf{w}}] = 0$, and minimizes the complexity

of the hypothesis $h_{\mathbf{w}}$ as measured by the norm of the weights.

If the problem is infeasible, the designer could add more basis functions $\mathbf{f}(\mathbf{x}, \mathbf{y})$ that take into account additional information about $\mathbf{x}$. One could also add slack variables for each example and each constraint that would allow violations of optimality conditions with some penalty. However, these slack variables would not represent upper bounds on the loss as they are in the min-max formulation, and therefore are less justified.

4.2 Approximations: upper and lower bounds

There are structured prediction tasks for which we might not be able to solve the estimation problem exactly. Often, we cannot compute $\max_{\mathbf{y} \in \mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$ exactly or explicitly, but can only upper or lower bound it. Fig. 4.1 shows schematically how approximating of the max subproblem reduces or extends the feasible space of $\mathbf{w}$ and ξ and leads to approximate solutions. The nature of these lower and upper bounds depends on the problem, but we consider two general cases below.

4.2.1 Constraint generation

When neither compact maximization or optimality formulation is possible, but the maximization problem can be solved or approximated by a combinatorial algorithm, we can resort to *constraint generation* or *cutting plane* methods. Consider Eq. (4.3), where we have an exponential number of linear constraints, one for each i and $\mathbf{y} \in \mathcal{Y}^{(i)}$. Only a subset of those constraints will be active at the optimal solution $\mathbf{w}$. In fact, not more than the number of parameters n plus the number of examples m can be active in general, since that is the number of variables. If we can identify a small number of constraints that are critical to the solution, we do not have to include all of them. Of course, identifying these constraints is in general as difficult as solving the problem, but a greedy approach of adding the most violated constraints often achieves good approximate solutions after adding a small (polynomial) number of constraints. If we continue adding constraints until there are no more violated ones, the resulting solution is optimal.

We assume that we have an algorithm that produces $\mathbf{y} = \arg\max_{\mathbf{y} \in \mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) +$

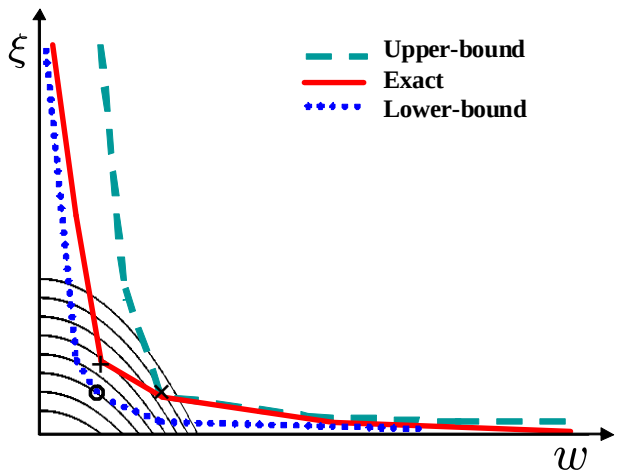


Figure 4.1: Exact and approximate constraints on the max-margin quadratic program. The solid red line represents the constraints imposed by the assignments $\mathbf{y} \in \mathbf{Y}^{(i)}$, whereas the dashed and dotted lines represent approximate constraints. The approximate constraints may coincide with the exact constraints in some cases, and be more stringent or relaxed in others. The parabolic contours represent the value of the objective function and '+', 'x' and 'o' mark the different optima.

$\ell_i(\mathbf{y})]$. The algorithm is described in Fig. 4.2. We maintain, for each example i, a small but growing set of assignments $\widetilde{\mathcal{Y}}^{(i)} \subset \mathcal{Y}^{(i)}$. At each iteration, we solve the problem with a subset of constraints:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y}), \quad \forall i, \quad \forall \mathbf{y} \in \widetilde{\mathcal{Y}}^{(i)}. \end{aligned} \tag{4.14}$$

The only difference between Eq. (4.3) and Eq. (4.14) is that $\mathcal{Y}^{(i)}$ has been replaced by $\widetilde{\mathcal{Y}}^{(i)}$. We then compute $\mathbf{y} = \arg\max_{\mathbf{y}\in\mathcal{Y}^{(i)}}[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$ for each i and check whether the constraint $\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i + \epsilon \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$, is violated, where ϵ is a user defined precision parameter. If it is violated, we set $\widetilde{\mathcal{Y}}^{(i)} = \widetilde{\mathcal{Y}}^{(i)} \cup \mathbf{y}$. The algorithm terminates when no constraints are violated. In Fig. 4.1, the lower-bound on the constraints provided by $\widetilde{\mathcal{Y}}^{(i)} \cup \mathbf{y}$ keeps tightening with each iteration, terminating when the desired precision ϵ is reached. We note that if the algorithm that produces $\mathbf{y} = \arg\max_{\mathbf{y}\in\mathcal{Y}^{(i)}}[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$ is

Input: precision parameter ϵ.

1. Initialize: $\widetilde{\mathcal{Y}}^{(i)} = \{\}, \ \ \forall\, i$.

2. Set $violation = 0$ and solve for $\mathbf{w}$ and ξ by optimizing

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y}), \quad \forall i, \ \ \forall \mathbf{y} \in \widetilde{\mathcal{Y}}^{(i)}. \end{aligned}$$

3. For each i,
   Compute $\mathbf{y} = \arg\max_{\mathbf{y}\in\mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$,
   if $\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i + \epsilon \leq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$,
   then set $\widetilde{\mathcal{Y}}^{(i)} = \widetilde{\mathcal{Y}}^{(i)} \cup \mathbf{y}$ and $violation = 1$

4. if $violation = 1$ goto 2.

Return $\mathbf{w}$.

Figure 4.2: A constraint generation algorithm.

suboptimal, the approximation error of the solution we achieve might be much greater than ϵ. The number of constraints that must be added before the algorithm terminates depends on the precision ϵ and problem specific characteristics. See [Bertsimas & Tsitsiklis, 1997; Boyd & Vandenberghe, 2004] for a more in-depth discussion of cutting planes methods. This approach may also be computationally faster in providing a very good approximation in practice if the explicit convex programming formulation is polynomial in size, but very large, while the maximization algorithm is comparatively fast.

4.2.2 Constraint strengthening

In many problems, the maximization problem we are interested in may be very expensive or intractable. For example, we consider MAP inference in large tree-width Markov networks in Ch. 8, multi-way cut in Ch. 7, graph-partitioning in Ch. 11. Many such problems can be written as *integer* programs. Relaxations of such integer programs into LPs, QPs or SDPs often provide excellent approximation algorithms [Hochbaum, 1997; Nemhauser & Wolsey, 1999]. The relaxation usually defines a larger feasible space $\widetilde{\mathcal{Y}}^{(i)} \supset \mathcal{Y}^{(i)}$ over

which the maximization is done, where $\mathbf{y} \in \widetilde{\mathcal{Y}}^{(i)}$ may correspond to a "fractional" assignment. For example, a solution to the MAP LP in Eq. (3.2) for an untriangulated network may not correspond to any valid assignment. In such a case, the approximation is an over-estimate of the constraints:

$$\max_{\mathbf{y} \in \widetilde{\mathcal{Y}}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})] \geq \max_{\mathbf{y} \in \mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})].$$

Hence the constraint set is tightened with such invalid assignments. Fig. 4.1 shows how the over-estimate reduces the feasible space of $\mathbf{w}$ and ξ.

Note that for every setting of the weights $\mathbf{w}$ that produces fractional solutions for the relaxation, the approximate constraints are tightened because of the additional invalid assignments. In this case, the approximate MAP solution has higher value than any integer solution, including the true assignment $\mathbf{y}^{(i)}$, thereby driving up the corresponding slack ξ_i. By contrast, for weights $\mathbf{w}$ for which the MAP approximation is integer-valued, the margin has the standard interpretation as the difference between the score of $\mathbf{y}^{(i)}$ and the MAP $\mathbf{y}$ (according to $\mathbf{w}$). As the objective includes a penalty for the slack variable, intuitively, minimizing the objective tends to drive the weights $\mathbf{w}$ away from the regions where the solutions to the approximation are fractional. In essence, the estimation algorithm is finding weights that are not necessarily optimal for an *exact* maximization algorithm, but (close to) optimal for the particular *approximate* maximization algorithm used. In practice, we will show experimentally that such approximations often work very well.

4.3 Related work

Our max-margin formulation is related to a body of work called inverse combinatorial and convex optimization [Burton & Toint, 1992; Zhang & Ma, 1996; Ahuja & Orlin, 2001; Heuberger, 2004]. An *inverse optimization problem* is defined by an instance of an optimization problem $\max_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\top \mathbf{f}(\mathbf{y})$, a set of nominal weights $\mathbf{w}^0$, and a target solution $\mathbf{y}^t$. The goal is to find the weights $\mathbf{w}$ closest to the nominal $\mathbf{w}^0$ in some norm, which make the

target solution optimal:

$$\begin{aligned} \min \quad & ||\mathbf{w} - \mathbf{w}^0||_p \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}(\mathbf{y}^t) \geq \mathbf{w}^\top \mathbf{f}(\mathbf{y}), \quad \forall \mathbf{y} \in \mathcal{Y}. \end{aligned}$$

Most of the attention has been on L_1 and L_∞ norms, but L_2 norm is also used.

The study of inverse problems began with geophysical scientists (see [Tarantola, 1987] for in-depth discussion of a wide range of applications). Modeling a complex physical system often involves a large number of parameters which scientists find hard or impossible to set correctly. Provided educated guesses for the parameters $\mathbf{w}^0$ and the behavior of the system as a target, the inverse optimization problem attempts to match the behavior while not perturbing the "guesstimate" too much.

Although there is a strong connection between inverse optimization problems and our formulations, the goals are very different than ours. In our framework, we are learning a parameterized objective function that depends on the input $\mathbf{x}$ and will generalize well in prediction on new instances. Moreover, we do not assume as given a nominal set of weights. Note that if we set $\mathbf{w}^0 = 0$, then $\mathbf{w} = 0$ is trivially the optimal solution. The solution $\mathbf{w}$ depends critically on the choice of nominal weights, which is not appropriate in the learning setting.

The inverse reinforcement learning problem [Ng & Russell, 2000; Abbeel & Ng, 2004] is much closer to our setting. The goal is to learn a reward function that will cause a rational agent to act similar to the observed behavior of an expert. A full description of the problem is beyond our scope, but we briefly describe the Markov decision process (MDP) model commonly used for sequential decision making problems where an agent interacts with its environment. The environment is modeled as a system that can be in one of a set of discrete states. At every time step, the agent chooses an action from a discrete set of actions and the system transitions to a next state with a probability that depends on the current state and the action taken. The agent collects a reward at each step, which generally depends on the on the current and the next state and the action taken. A rational agent executes a policy (essentially, a state to action mapping) that maximizes its expected reward. To map this problem (approximately) to our setting, note that a policy roughly corresponds to the labels

$\mathbf{y}$, the state sequence correspond to the input $\mathbf{x}$ and the reward for a state/action sequence is assumed to be $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$ for some basis functions $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$. The goal is to learn $\mathbf{w}$ from a set of state/action sequences $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ of the expert such that the maximizing the expected reward according to the system model makes the agent imitate the expert. This and related problems are formulated as a convex program in Ng and Russell [2000] and Abbeel and Ng [2004].

4.4 Conclusion

In this chapter, we presented two formulations of structured max-margin estimation that define a compact convex optimization problem. The first formulation, **min-max**, relies on the ability to express inference in the model as a compact convex optimization problem. The second one, **certificate**, only requires expressing optimality of a given assignment according to the model. Our framework applies to a wide range of prediction problems that we explore in the rest of the thesis, including Markov networks, context free grammars, and many combinatorial structures such as matchings and graph-cuts. The estimation problem is tractable and exact whenever the prediction problem can be formulated as a compact convex optimization problem or a polynomial time combinatorial algorithm with compact convex optimality conditions. When the prediction problem is intractable or very expensive to solve exactly, we resort to approximations that only provide upper/lower bounds on the predictions. The estimated parameters are then approximate, but produce accurate *approximate* prediction models in practice.

Because our approach only relies using the maximum in the model for prediction, and does not require a normalized distribution $P(\mathbf{y} \mid \mathbf{x})$ over all outputs, maximum margin estimation can be tractable when maximum likelihood is not. For example, to learn a probabilistic model $P(\mathbf{y} \mid \mathbf{x})$ over bipartite matchings using maximum likelihood requires computing the normalizing partition function, which is #P-complete [Valiant, 1979; Garey & Johnson, 1979]. By contrast, maximum margin estimation can be formulated as a compact QP with linear constraints. Similar results hold for non-bipartite matchings and min-cuts.

In models that are tractable for both maximum likelihood and maximum margin, (such as low-treewidth Markov networks, context free grammars, many other problems in which

inference is solvable by dynamic programming), our approach has an additional advantage. Because of the hinge-loss, the solutions to the estimation are relatively sparse in the dual space (as in SVMs), which makes the use of kernels much more efficient. Maximum likelihood estimation with kernels results in models that are generally non-sparse and require pruning or greedy support vector selection methods [Lafferty *et al.*, 2004; Altun *et al.*, 2004].

The forthcoming formulations in the thesis follow the principles laid out in this chapter. The range of applications of these principles is very broad and leads to estimation problems with very interesting structure in each particular problem, from Markov networks and context-free grammars to graph cuts and perfect matchings.

Part II

Markov networks

Chapter 5

Markov networks

Markov networks are extensively used to model complex sequential, spatial, and relational interactions in prediction problems arising in many fields. These problems involve labeling a set of related objects that exhibit *local* consistency. In *sequential* labeling problems (such as handwriting recognition), the labels (letters) of adjacent inputs (images) are highly correlated. Sequential prediction problems arise in natural language processing (part-of-speech tagging, speech recognition, information extraction [Manning & Schütze, 1999]), computational biology (gene finding, protein structure prediction, sequence alignment [Durbin *et al.*, 1998]), and many other fields. In image processing, neighboring pixels exhibit *spatial* label coherence in denoising, segmentation and stereo correspondence [Besag, 1986; Boykov *et al.*, 1999a]. In hypertext or bibliographic classification, labels of linked and co-cited documents tend to be similar [Chakrabarti *et al.*, 1998; Taskar *et al.*, 2002]. In proteomic analysis, location and function of proteins that interact are often highly correlated [Vazquez *et al.*, 2003]. Markov networks compactly represent complex joint distributions of the label variables by modeling their local interactions. Such models are encoded by a graph, whose nodes represent the different object labels, and whose edges represent and quantify direct dependencies between them. For example, a Markov network for the hypertext domain would include a node for each webpage, encoding its label, and an edge between any pair of webpages whose labels are directly correlated (e.g., because one links to the other).

We address the problem of max-margin estimation the parameters of Markov networks

for such structured classification problems. We show a compact convex formulation that seamlessly integrates kernels with graphical models. We analyze the theoretical generalization properties of max-margin estimation and derive a novel margin-based bound for structured classification.

We are given a labeled training sample $S = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{m}$, drawn from a fixed distribution D over $\mathcal{X} \times \mathcal{Y}$. We assume the structure of the network is given: we have a mapping from an input $\mathbf{x}$ to the corresponding Markov network graph $\mathcal{G}(\mathbf{x}) = \{\mathcal{V}, \mathcal{E}\}$ where the nodes $\mathcal{V}$ map to the variables in $\mathbf{y}$. We abbreviate $\mathcal{G}(\mathbf{x}^{(i)})$ as $\mathcal{G}^{(i)}$ below. In handwriting recognition, this mapping depends on the segmentation algorithm that determines how many letters the sample image contains and splits the image into individual images for each letter. It also depends on the basis functions we use to model the dependencies of the problem, for example, first-order Markov chain or a higher-order models. Note that the topology and size of the graph $\mathcal{G}^{(i)}$, might be different for each example i. For instance, the training sequences might have different lengths.

We focus on *conditional* Markov networks (or CRFs [Lafferty *et al.*, 2001]), which represent $P(\mathbf{y} \mid \mathbf{x})$ instead of *generative* models $P(\mathbf{x}, \mathbf{y})$. The log-linear representation we have described in Sec. 3.3.1 is defined via a vector of n basis functions $\mathbf{f}(\mathbf{x}, \mathbf{y})$:

$$\log P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x}) = \mathbf{w}^{\top}\mathbf{f}(\mathbf{x}, \mathbf{y}) - \log Z_{\mathbf{w}}(\mathbf{x}),$$

where $Z_{\mathbf{w}}(\mathbf{x}) = \sum_{\mathbf{y}} \exp\{\mathbf{w}^{\top}\mathbf{f}(\mathbf{x}, \mathbf{y})\}$ and $\mathbf{w} \in \mathbb{R}^n$. Before we present the maximum margin estimation, we review the standard maximum likelihood method.

5.1 Maximum likelihood estimation

The regularized maximum likelihood approach of learning the weights $\mathbf{w}$ of a Markov network is similar to logistic regression we described in Sec. 2.2. The objective is to minimize the training log-loss with an additional regularization term, usually the squared-norm of the weights $\mathbf{w}$ [Lafferty *et al.*, 2001]:

$$\frac{1}{2}||\mathbf{w}||^2 - C\sum_i \log P_{\mathbf{w}}(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)}) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \log Z_{\mathbf{w}}(\mathbf{x}^{(i)}) - \mathbf{w}^{\top}\mathbf{f}_i(\mathbf{y}^{(i)}),$$

where $\mathbf{f}_i(\mathbf{y}) = \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y})$.

This objective function is convex in the parameters $\mathbf{w}$, so we have an unconstrained convex optimization problem. The gradient with respect to $\mathbf{w}$ is given by:

$$\mathbf{w} + C \sum_i \left[\mathbf{E}_{i,\mathbf{w}}[\mathbf{f}_i(\mathbf{y})] - \mathbf{f}_i(\mathbf{y}^{(i)})\right] = \mathbf{w} - C \sum_i \mathbf{E}_{i,\mathbf{w}}[\Delta \mathbf{f}_i(\mathbf{y})],$$

where $\mathbf{E}_{i,\mathbf{w}}[\mathbf{f}_i(\mathbf{y})] = \sum_{\mathbf{y} \in \mathcal{Y}} \mathbf{f}_i(\mathbf{y}) P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x}^{(i)})$ is the expectation under the conditional distribution $P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x}^{(i)})$ and $\Delta \mathbf{f}_i(\mathbf{y}) = \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y})$, as before. To compute the expectations, we can use inference in the Markov network to calculate marginals $P_{\mathbf{w}}(\mathbf{y}_c \mid \mathbf{x}^{(i)})$ for each clique c in the network Sec. 3.3.2. Since the basis functions decompose over the cliques of the network, the expectation decomposes as well:

$$\mathbf{E}_{i,\mathbf{w}}[\mathbf{f}_i(\mathbf{y})] = \sum_{c \in \mathcal{C}^{(i)}} \sum_{\mathbf{y}_c \in \mathcal{Y}_c^{(i)}} \mathbf{f}_{i,c}(\mathbf{y}_c) P_{\mathbf{w}}(\mathbf{y}_c \mid \mathbf{x}^{(i)}).$$

Second order methods for solving unconstrained convex optimization problems, such as Newton's method, require the second derivatives as well as the gradient. Let $\delta \mathbf{f}_i(\mathbf{y}) = \mathbf{f}_i(\mathbf{y}) - \mathbf{E}_{i,\mathbf{w}}[\mathbf{f}_i(\mathbf{y})]$. The Hessian of the objective depends on the covariances of the basis functions:

$$I + C \sum_i \mathbf{E}_{i,\mathbf{w}} \left[\delta \mathbf{f}_i(\mathbf{y}) \delta \mathbf{f}_i(\mathbf{y})^\top\right],$$

where I is a $n \times n$ identity matrix. Computing the Hessian is more expensive than the gradient, since we need to calculate joint marginals of every pair of cliques c and c', $P_{\mathbf{w}}(\mathbf{y}_{c \cup c'} \mid \mathbf{x}_i)$ as well as covariances of all basis functions, which is quadratic in the number of cliques and the number of functions. A standard approach is to use an approximate second order method that does not need to compute the Hessian, but uses only the gradient information [Nocedal & Wright, 1999; Boyd & Vandenberghe, 2004]. Conjugate Gradients or L-BFGS methods have been shown to work very well on large estimation problems [Sha & Pereira, 2003; Pinto *et al.*, 2003], even with millions of parameters $\mathbf{w}$.

5.2 Maximum margin estimation

For maximum-margin estimation, we begin with the min-max formulation from Sec. 4.1:

$$\begin{aligned} &\min \ \ \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ &\text{s.t.} \ \ \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq \max_{\mathbf{y}} \ [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})], \quad \forall i. \end{aligned} \tag{5.1}$$

We know from Sec. 3.3.3 how to express $\max_{\mathbf{y}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$ as an LP, but the important difference is the loss function ℓ_i. The simplest loss is the $0/1$ loss $\ell_i(\mathbf{y}) \equiv \mathbb{1}(\mathbf{y}^{(i)} \neq \mathbf{y})$. In fact this loss for sequence models was used by Collins [2001] and Altun *et al.* [2003]. However, in structured problems, where we are predicting multiple labels, the loss is often not just the simple $0/1$ loss, but may depend on the number of labels and type of labels predicted incorrectly or perhaps the number of cliques of labels predicted incorrectly. In general, we assume that the loss, like the basis functions, decomposes over the cliques of labels.

Assumption 5.2.1 *The loss function $\ell_i(\mathbf{y})$ is decomposable:*

$$\ell_i(\mathbf{y}) = \sum_{c \in \mathcal{C}(\mathcal{G}^{(i)})} \ell(\mathbf{x}_c^{(i)}, \mathbf{y}_c^{(i)}, \mathbf{y}_c) = \sum_{c \in \mathcal{C}(\mathcal{G}^{(i)})} \ell_{i,c}(\mathbf{y}_c).$$

We will focus on decomposable loss functions below. A natural choice that we use in our experiments is the Hamming distance:

$$\ell^H(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{y}) = \sum_{v \in \mathcal{V}^{(i)}} \mathbb{1}(y_v^{(i)} \neq y_v).$$

With this assumption, we can express this inference problem for a triangulated graph as a linear program for each example i as in Sec. 3.3.3:

$$\begin{aligned} \max \quad & \sum_{c, \mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c)[\mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c) + \ell_{i,c}(\mathbf{y}_c)] \\ \text{s.t.} \quad & \sum_{\mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c) = 1, \quad \forall i, \ \ \forall c \in \mathcal{C}^{(i)}; \qquad \mu_{i,c}(\mathbf{y}_c) \geq 0, \ \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c; \\ & \mu_{i,s}(\mathbf{y}_s) = \sum_{\mathbf{y}'_c \sim \mathbf{y}_s} \mu_{i,c}(\mathbf{y}'_c), \qquad \forall s, c \in \mathcal{C}^{(i)}, \ \ s \subset c, \ \ \forall \mathbf{y}_s, \end{aligned} \tag{5.2}$$

where $\mathcal{C}^{(i)} = \mathcal{C}(\mathcal{G}^{(i)})$ are the cliques of the Markov network for example i.

As we showed before, the constraints ensure that the μ_i's form a proper distribution. If the most likely assignment is unique, then the distribution that maximizes the objective puts all its weight on that assignment. (If the $\arg\max$ is not unique, any convex combination of the assignments is a valid solution). The dual of Eq. (5.2) is given by:

$$\begin{aligned} &\min \sum_c \lambda_{i,c} \\ &\text{s.t.} \quad \lambda_{i,c} + \sum_{s \supset c} m_{i,s,c}(\mathbf{y}_c) - \sum_{s \subset c,\, \mathbf{y}'_s \sim \mathbf{y}_c} m_{i,c,s}(\mathbf{y}'_s) \geq \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c) + \ell_{i,c}(\mathbf{y}_c), \quad \forall c \in \mathcal{C}^{(i)}, \forall \mathbf{y}_c. \end{aligned} \tag{5.3}$$

In this dual, the $\lambda_{i,c}$ variables correspond to the normalization constraints, while $m_{i,c,s}(\mathbf{y}_c)$ variables correspond to the agreement constraints in the primal in Eq. (5.2).

Plugging the dual into Eq. (5.1) for each example i and maximizing jointly over all the variables ($\mathbf{w}, \xi, \lambda$ and $\mathbf{m}$), we have:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_i \xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq \sum_{i,c} \lambda_{i,c}, \quad \forall i; \\ & \lambda_{i,c} + \sum_{s \supset c} m_{i,s,c}(\mathbf{y}_c) - \sum_{s \subset c,\, \mathbf{y}'_s \sim \mathbf{y}_c} m_{i,c,s}(\mathbf{y}'_s) \geq \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c) + \ell_{i,c}(\mathbf{y}_c), \quad \forall c \in \mathcal{C}^{(i)}, \forall \mathbf{y}_c. \end{aligned} \tag{5.4}$$

In order to gain some intuition about this formulation, we make a change of variables from $\lambda_{i,c}$ to $\xi_{i,c}$:

$$\lambda_{i,c} = \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c^{(i)}) + \xi_{i,c}, \quad \forall i, \quad \forall c \in \mathcal{C}^{(i)}.$$

The reason for naming the new variables using the letter ξ will be clear in the following. For readability, we also introduce variables that capture the effect of all the agreement variables $\mathbf{m}$:

$$M_{i,c}(\mathbf{y}_c) = \sum_{s \subset c,\, \mathbf{y}'_s \sim \mathbf{y}_c} m_{i,c,s}(\mathbf{y}'_s) - \sum_{s \supset c} m_{i,s,c}(\mathbf{y}_c), \quad \forall i, \quad \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c.$$

With these new variables, we have:

$$\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\
\text{s.t.} \quad & \xi_i \geq \sum_c \xi_{i,c}, \quad \forall i; \\
& \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c^{(i)}) + \xi_{i,c} \geq \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c) + \ell_{i,c}(\mathbf{y}_c) + M_{i,c}(\mathbf{y}_c), \quad \forall i, \quad \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c; \\
& M_{i,c}(\mathbf{y}_c) = \sum_{s \subset c,\ \mathbf{y}'_s \sim \mathbf{y}_c} m_{i,c,s}(\mathbf{y}'_s) - \sum_{s \supset c} m_{i,s,c}(\mathbf{y}_c), \quad \forall i, \quad \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c.
\end{aligned} \tag{5.5}$$

Note that $\xi_i = \sum_c \xi_{i,c}$ at the optimum, since the slack variable ξ_i only appears only in the constraint $\xi_i \geq \sum_c \xi_{i,c}$ and the objective minimizes $C\xi_i$. Hence we can simply eliminate this set of variables:

$$\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i,c} \xi_{i,c} \\
\text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c^{(i)}) + \xi_{i,c} \geq \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c) + \ell_{i,c}(\mathbf{y}_c) + M_{i,c}(\mathbf{y}_c), \quad \forall i, \quad \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c; \\
& M_{i,c}(\mathbf{y}_c) = \sum_{s \subset c,\ \mathbf{y}'_s \sim \mathbf{y}_c} m_{i,c,s}(\mathbf{y}'_s) - \sum_{s \supset c} m_{i,s,c}(\mathbf{y}_c), \quad \forall i, \quad \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c.
\end{aligned} \tag{5.6}$$

Finally, we can write this in a form that resembles our original formulation Eq. (5.1), but defined at a local level, for each clique:

$$\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i,c} \xi_{i,c} \\
\text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c^{(i)}) + \xi_{i,c} \geq \max_{\mathbf{y}_c} \left[\mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c) + \ell_{i,c}(\mathbf{y}_c) + M_{i,c}(\mathbf{y}_c)\right], \quad \forall i, \quad \forall c \in \mathcal{C}^{(i)}; \\
& M_{i,c}(\mathbf{y}_c) = \sum_{s \subset c,\ \mathbf{y}'_s \sim \mathbf{y}_c} m_{i,c,s}(\mathbf{y}'_s) - \sum_{s \supset c} m_{i,s,c}(\mathbf{y}_c), \quad \forall i, \quad \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c.
\end{aligned} \tag{5.7}$$

Note that without $M_{i,c}$ and $m_{i,c,s}$ variables, we essentially treat each clique as an independent classification problem: for each clique we have a hinge upper-bound on the local loss, or a margin requirement. The $m_{i,c,s}(\mathbf{y}_s)$ variables correspond to a certain kind of messages between cliques that distribute "credit" to cliques to fulfill this margin requirement from other cliques which have sufficient margin.

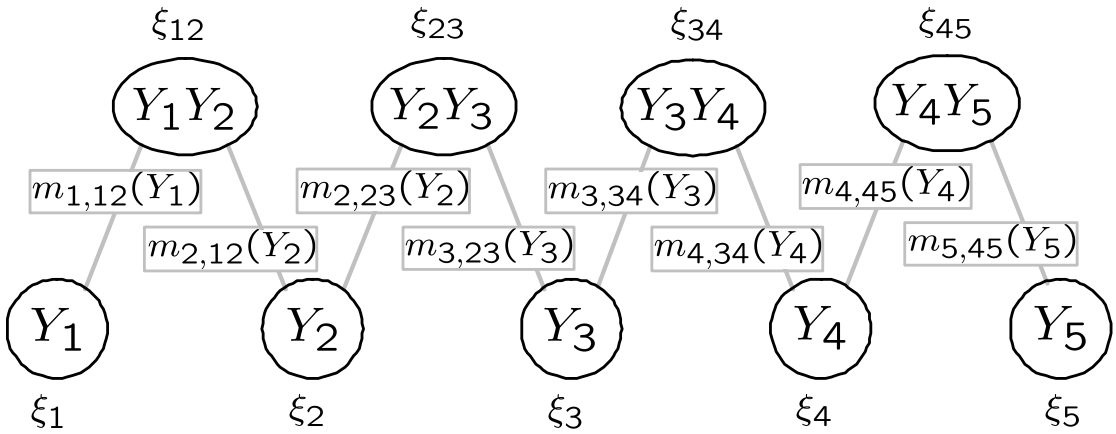


Figure 5.1: First-order chain shown as a set of cliques (nodes and edges). Also shown are the corresponding local slack variables ξ for each clique and messages m between cliques.

As an example, consider the first-order Markov chain in Fig. 5.1. The set of cliques consists of the five nodes and the four edges. Suppose for the sake of this example that our training data consists of only one training sample. The figure shows the local slack variables ξ and messages m between cliques for this sample. For brevity of notion in this example, we drop the dependence on the sample index i in the indexing of the variables (we also used $y_j^{(*)}$ instead of $y_j^{(i)}$ below). For concreteness, below we use the Hamming loss ℓ^H, which decomposes into local terms $\ell_j(y_j) = \mathbb{1}(y_j \neq y_j^{(*)})$ for each node and is zero for the edges.

The constraints associated with the node cliques in this sequence are:

$$
\begin{aligned}
\mathbf{w}^\top \mathbf{f}_1(y_1^{(*)}) + \xi_1 &\geq \mathbf{w}^\top \mathbf{f}_1(y_1) + \mathbb{1}(y_1 \neq y_1^{(*)}) - m_{1,12}(y_1), \quad \forall y_1; \\
\mathbf{w}^\top \mathbf{f}_2(y_2^{(*)}) + \xi_2 &\geq \mathbf{w}^\top \mathbf{f}_2(y_2) + \mathbb{1}(y_2 \neq y_2^{(*)}) - m_{2,12}(y_2) - m_{2,23}(y_2), \quad \forall y_2; \\
\mathbf{w}^\top \mathbf{f}_3(y_3^{(*)}) + \xi_3 &\geq \mathbf{w}^\top \mathbf{f}_3(y_3) + \mathbb{1}(y_3 \neq y_3^{(*)}) - m_{3,23}(y_3) - m_{3,34}(y_3), \quad \forall y_3; \\
\mathbf{w}^\top \mathbf{f}_4(y_4^{(*)}) + \xi_4 &\geq \mathbf{w}^\top \mathbf{f}_4(y_4) + \mathbb{1}(y_4 \neq y_4^{(*)}) - m_{4,34}(y_4) - m_{4,45}(y_4), \quad \forall y_4; \\
\mathbf{w}^\top \mathbf{f}_5(y_5^{(*)}) + \xi_5 &\geq \mathbf{w}^\top \mathbf{f}_5(y_5) + \mathbb{1}(y_5 \neq y_5^{(*)}) - m_{5,45}(y_5), \quad \forall y_5.
\end{aligned}
$$

The edge constraints are:

$$\begin{aligned}
\mathbf{w}^\top \mathbf{f}_{12}(y_1^{(*)}, y_2^{(*)}) + \xi_{12} &\geq \mathbf{w}^\top \mathbf{f}_{12}(y_1, y_2) + m_{1,12}(y_1) + m_{2,12}(y_2), \quad \forall y_1, y_2; \\
\mathbf{w}^\top \mathbf{f}_{23}(y_2^{(*)}, y_3^{(*)}) + \xi_{23} &\geq \mathbf{w}^\top \mathbf{f}_{23}(y_2, y_3) + m_{2,23}(y_2) + m_{3,23}(y_3), \quad \forall y_2, y_3; \\
\mathbf{w}^\top \mathbf{f}_{34}(y_3^{(*)}, y_4^{(*)}) + \xi_{34} &\geq \mathbf{w}^\top \mathbf{f}_{34}(y_3, y_4) + m_{3,34}(y_3) + m_{4,34}(y_4), \quad \forall y_3, y_4; \\
\mathbf{w}^\top \mathbf{f}_{45}(y_4^{(*)}, y_5^{(*)}) + \xi_{45} &\geq \mathbf{w}^\top \mathbf{f}_{45}(y_4, y_5) + m_{4,45}(y_4) + m_{5,45}(y_5), \quad \forall y_4, y_5.
\end{aligned}$$

5.3 M^{3}N dual and kernels

In the previous section, we showed a derivation of a compact formulation based on LP inference. In this section, we develop an alternative dual derivation that provides a very interesting interpretation of the problem and is a departure for special-purpose algorithms we develop. We begin with the formulation as in Eq. (4.3):

$$\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_i \xi_i \\
\text{s.t.} \quad & \mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y}) \geq \ell_i(\mathbf{y}) - \xi_i, \ \forall i, \mathbf{y},
\end{aligned} \tag{5.8}$$

where $\Delta \mathbf{f}_i(y) \equiv \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y})$. The dual is given by:

$$\begin{aligned}
\max \quad & \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y}) \ell_i(\mathbf{y}) - \frac{1}{2} \left|\left| \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y}) \Delta \mathbf{f}_i(\mathbf{y}) \right|\right|^2 \\
\text{s.t.} \quad & \sum_{\mathbf{y}} \alpha_i(\mathbf{y}) = C, \ \forall i; \qquad \alpha_i(\mathbf{y}) \geq 0, \ \forall i, \mathbf{y}.
\end{aligned} \tag{5.9}$$

In the dual, the exponential number of $\alpha_i(\mathbf{y})$ variables correspond to the exponential number of constraints in the primal. We make two small transformations to the dual that do not change the problem: we normalize α's by C (by letting $\alpha_i(\mathbf{y}) = C\alpha'_i(\mathbf{y})$), so that they sum

to 1 and divide the objective by C. The resulting dual is given by:

$$\begin{aligned} \max \quad & \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})\ell_i(\mathbf{y}) - \frac{1}{2}C\left|\left|\sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y})\right|\right|^2 \\ \text{s.t.} \quad & \sum_{\mathbf{y}} \alpha_i(\mathbf{y}) = 1,\ \forall i; \qquad \alpha_i(\mathbf{y}) \geq 0,\ \ \forall i, \mathbf{y}. \end{aligned} \tag{5.10}$$

As in multi-class SVMs, the solution to the dual α gives the solution to the primal as a weighted combination: $\mathbf{w}^* = C\sum_{i,\mathbf{y}} \alpha_i^*(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y})$.

Our main insight is that the variables $\alpha_i(\mathbf{y})$ in the dual formulation Eq. (5.10) can be interpreted as a kind of *distribution* over $\mathbf{y}$, since they lie in the simplex

$$\sum_{\mathbf{y}} \alpha_i(\mathbf{y}) = 1; \quad \alpha_i(\mathbf{y}) \geq 0,\ \forall \mathbf{y}.$$

This dual distribution does not represent the probability that the model assigns to an instantiation, but the importance of the constraint associated with the instantiation to the solution. The dual objective is a function of expectations of $\ell_i(\mathbf{y})$ and $\Delta\mathbf{f}_i(\mathbf{y})$ with respect to $\alpha_i(\mathbf{y})$. Since $\ell_i(\mathbf{y}) = \sum_c \ell_{i,c}(\mathbf{y}_c)$ and $\Delta\mathbf{f}_i(\mathbf{y}) = \sum_c \Delta\mathbf{f}_{i,c}(\mathbf{y}_c)$ decompose over the cliques of the Markov network, we only need clique marginals of the distribution $\alpha_i(\mathbf{y})$ to compute their expectations. We define the marginal dual variables as follows:

$$\mu_{i,c}(\mathbf{y}_c) = \sum_{\mathbf{y}' \sim \mathbf{y}_c} \alpha_i(\mathbf{y}'), \qquad \forall i, \quad \forall c \in \mathcal{C}^{(i)}, \quad \forall \mathbf{y}_c, \tag{5.11}$$

where $\mathbf{y}' \sim \mathbf{y}_c$ denotes whether the partial assignment $\mathbf{y}_c$ is consistent with the full assignment $\mathbf{y}'$. Note that the number of $\mu_{i,c}(\mathbf{y}_c)$ variables is small (polynomial) compared to the number of $\alpha_i(\mathbf{y})$ variables (exponential) if the size of the largest clique is constant with respect to the size of the network.

Now we can reformulate our entire QP (5.10) in terms of these marginal dual variables. Consider, for example, the first term in the objective function (fixing a particular i):

$$\sum_{\mathbf{y}} \alpha_i(\mathbf{y})\ell_i(\mathbf{y}) = \sum_{\mathbf{y}} \alpha_i(\mathbf{y}) \sum_c \ell_{i,c}(\mathbf{y}_c) = \sum_{c,\mathbf{y}_c} \ell_{i,c}(\mathbf{y}_c) \sum_{\mathbf{y}' \sim \mathbf{y}_c} \alpha_i(\mathbf{y}') = \sum_{c,\mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c)\ell_{i,c}(\mathbf{y}_c).$$

The decomposition of the second term in the objective is analogous.

$$\sum_{\mathbf{y}} \alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y}) = \sum_{c,\mathbf{y}_c} \Delta\mathbf{f}_{i,c}(\mathbf{y}_c) \sum_{\mathbf{y}'\sim\mathbf{y}_c} \alpha_i(\mathbf{y}') = \sum_{c,\mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c)\Delta\mathbf{f}_{i,c}(\mathbf{y}_c).$$

Let us denote the the objective of Eq. (5.10) as $\mathcal{Q}(\alpha)$. Note that it only depends on $\alpha_i(\mathbf{y})$ through its marginals $\mu_{i,c}(\mathbf{y}_c)$, that is, $\mathcal{Q}(\alpha) = \mathcal{Q}'(\mathcal{M}(\alpha))$, where $\mathcal{M}$ denotes the marginalization operator defined by Eq. (5.11) . The domain of this operator, $\mathcal{D}[\mathcal{M}]$, is the product of simplices for all the m examples. What is its range, $\mathcal{R}[\mathcal{M}]$, the set of legal marginals? Characterizing this set (also known as *marginal polytope*) compactly will allow us to work in the space of μ's:

$$\max_{\alpha\in\mathcal{D}[\mathcal{M}]} \mathcal{Q}(\alpha) \Leftrightarrow \max_{\mu\in\mathcal{R}[\mathcal{M}]} \mathcal{Q}'(\mu).$$

Hence we must ensure that μ_i corresponds to *some* distribution α_i, which is exactly what the constraints in the LP for MAP inference enforce (see discussion of Lemma 3.3.5). Therefore, when all $\mathcal{G}^{(i)}$ are triangulated, the following *structured* dual QP has the same primal solution ($\mathbf{w}^*$) as the original *exponential* dual QP in Eq. (5.10):

$$\begin{aligned}
\max \quad & \sum_{i,c,\mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c)\ell_{i,c}(\mathbf{y}_c) - \frac{1}{2}C\left|\left|\sum_{i,c,\mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c)\Delta\mathbf{f}_{i,c}(\mathbf{y}_c)\right|\right|^2 \\
\text{s.t.} \quad & \sum_{\mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c) = 1, \quad \forall i,\ \forall c\in\mathcal{C}^{(i)}; \qquad \mu_{i,c}(\mathbf{y}_c) \geq 0,\ \forall i,\ \forall c\in\mathcal{C}^{(i)},\ \forall \mathbf{y}_c; \\
& \mu_{i,s}(\mathbf{y}_s) = \sum_{\mathbf{y}'_c\sim\mathbf{y}_s} \mu_{i,c}(\mathbf{y}'_c), \quad \forall i,\ \forall s,c\in\mathcal{C}^{(i)},\ s\subset c,\ \forall \mathbf{y}_s.
\end{aligned} \tag{5.12}$$

The solution to the structured dual μ^* gives us the primal solution:

$$\mathbf{w}^* = C\sum_{i,c,\mathbf{y}_c} \mu^*_{i,c}(\mathbf{y}_c)\Delta\mathbf{f}_{i,c}(\mathbf{y}_c).$$

In this structured dual, we only enforce that there exists an α_i consistent with μ_i, but do not make a commitment about what it is. In general, the α distribution is not unique, but there is a continuum of distributions consistent with a set of marginals. The objective of

the QP Eq. (5.10) does not distinguish between these distributions, since it only depends on their marginals. The maximum-entropy distribution α_i consistent with a set of marginals μ_i, however, is unique for a triangulated model and can be computed using the junction tree $\mathcal{T}^{(i)}$ for the network [Cowell *et al.*, 1999].

Specifically, associated with each edge (c, c') in the tree $\mathcal{T}^{(i)}$ is a set of variables called the separator $s = c \cap c'$. Note that each separator s and complement of a separator $c \setminus s$ is also a clique of the original graph, since it is a subclique of a larger clique. We denote the set of separators as $\mathcal{S}^{(i)}$. Now we can define the maximum-entropy distribution $\alpha_i(\mathbf{y})$ as follows:

$$\alpha_i(\mathbf{y}) = \frac{\prod_{c \in \mathcal{T}^{(i)}} \mu_{i,c}(\mathbf{y}_c)}{\prod_{s \in \mathcal{S}^{(i)}} \mu_{i,s}(\mathbf{y}_s)}. \tag{5.13}$$

Again, by convention $0/0 \equiv 0$.

Kernels

Note that the solution is a weighted combination of local basis functions and the objective of Eq. (5.12) can be expressed in terms of dot products between local basis functions

$$\Delta\mathbf{f}_{i,c}(\mathbf{y}_c)^\top \Delta\mathbf{f}_{j,\bar{c}}(\mathbf{y}_{\bar{c}}) = [\mathbf{f}(\mathbf{x}_c^{(i)}, \mathbf{y}_c^{(i)}) - \mathbf{f}(\mathbf{x}_c^{(i)}, \mathbf{y}_c)]^\top [\mathbf{f}(\mathbf{x}_{\bar{c}}^{(j)}, \mathbf{y}_{\bar{c}}^{(j)}) - \mathbf{f}(\mathbf{x}_{\bar{c}}^{(j)}, \mathbf{y}_{\bar{c}})].$$

Hence, we can locally kernelize our models and solve Eq. (5.12) efficiently. Kernels are typically defined on the input, e.g. $k(\mathbf{x}_c^{(i)}, \mathbf{x}_{\bar{c}}^{(j)})$. In our handwriting example, we use a polynomial kernel on the pixel values for the node cliques. We usually extend the kernel over the input space to the joint input and output space by simply defining

$$\mathbf{f}(\mathbf{x}_c, \mathbf{y}_c)^\top \mathbf{f}(\mathbf{x}_{\bar{c}}, \mathbf{y}_{\bar{c}}) \equiv \mathbb{1}(\mathbf{y}_c = \mathbf{y}_{\bar{c}}) k(\mathbf{x}_c, \mathbf{x}_{\bar{c}}).$$

Of course, other definitions are possible and may be useful when the assignments in each clique $\mathbf{y}_c$ have interesting structure. In Sec. 6.2 we experiment with several kernels for the handwriting example. As in SVMs, the solutions to the max-margin QP are typically sparse in the μ variables. Hence, each log-potential in the network "remembers" only a small proportion of the relevant training data inputs.

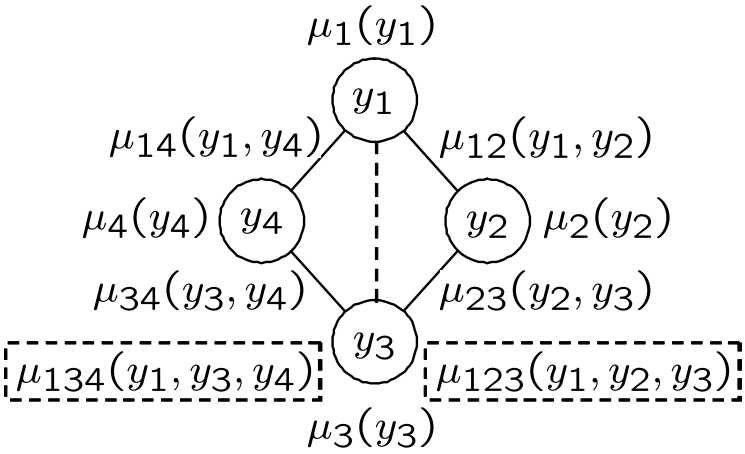


Figure 5.2: Diamond Markov network (added triangulation edge is dashed and three-node marginals are in dashed rectangles).

5.4 Untriangulated models

If the underlying Markov net is not chordal, we must address the problem by triangulating the graph, that is, adding fill-in edges to ensure triangulation. For example, if our graph is a 4-cycle Y_1—Y_2—Y_3—Y_4—Y_1 as in Fig. 5.2, we can triangulate the graph by adding an arc Y_1—Y_3. This will introduce new cliques Y_1, Y_2, Y_3 and Y_1, Y_3, Y_4 and the corresponding marginals, $\mu_{123}(y_1, y_2, y_3)$ and $\mu_{134}(y_1, y_3, y_4)$. We can then use this new graph to produce the constraints on the marginals:

$$\begin{aligned}
\sum_{y_1} \mu_{123}(y_1, y_2, y_3) &= \mu_{23}(y_2, y_3), && \forall y_2, y_3; \\
\sum_{y_3} \mu_{123}(y_1, y_2, y_3) &= \mu_{12}(y_1, y_2), && \forall y_1, y_2; \\
\sum_{y_1} \mu_{134}(y_1, y_3, y_4) &= \mu_{34}(y_3, y_4), && \forall y_3, y_4; \\
\sum_{y_3} \mu_{134}(y_1, y_3, y_4) &= \mu_{13}(y_1, y_3), && \forall y_1, y_3.
\end{aligned}$$

The new marginal variables appear only in the constraints; they do not add any new basis functions nor change the objective function.

In general, the number of constraints introduced is exponential in the number of variables in the new cliques — the tree-width of the graph. Unfortunately, even sparsely connected networks, for example 2D grids often used in image analysis, have large tree-width. However, we can still solve the QP in the structured primal Eq. (5.6) or the structured dual Eq. (5.12) defined by an untriangulated graph. Such a formulation, which enforces only local consistency of marginals, optimizes our objective only over a relaxation of the marginal polytope. However, the learned parameters produce very accurate approximate models in practice, as experiments in Ch. 8 demonstrate.

Note that we could also strengthen the untriangulated relaxation without introducing an exponential number of constraints. For example, we can add positive semidefinite constraints on the marginals μ used by Wainwright and Jordan [2003], which tend to improve the approximation of the marginal polytope. Although this and other more complex relaxations are a very interesting area of future development, they are often much more expensive.

The approximate QP does not guarantee that the learned model using *exact* inference minimizes the true objective: (upper-bound on) empirical risk plus regularization. But do we really need these optimal parameters if we cannot perform exact inference? A more useful goal is to make sure that training error is minimized using the *approximate* inference procedure via the untriangulated LP. We conjecture that the parameters learned by the approximate QP in fact do that to some degree. For instance, consider the separable case, where 100% accuracy is achievable on the training data by some parameter setting $\mathbf{w}$ such that approximate inference (using the untriangulated LP) produces integral solutions. Solving the problem as $C \to \infty$ will find this solution even though it may not be optimal (in terms of the norm of the $\mathbf{w}$) using exact inference. For C in intermediate range, the formulation trades off fractionality of the untriangulated LP solutions with complexity of the weights $||\mathbf{w}||^2$.

5.5 Generalization bound

In this section, we show a generalization bound for the task of structured classification that allows us to relate the error rate on the training set to the generalization error. To the best

of our knowledge, this bound is the first to deal with structured error, such as the Hamming distance. Our analysis of Hamming loss allows to prove a significantly stronger result than previous bounds for the $0/1$ loss, as we detail below.

Our goal in structured classification is often to minimize the number of misclassified labels, or the Hamming distance between $\mathbf{y}$ and $h(\mathbf{x})$. An appropriate error function is the *average per-label loss*

$$\mathcal{L}(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \frac{1}{L}\ell^H(\mathbf{y}, \arg\max_{\mathbf{y}'} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}')),$$

where L is the number of label variables in $\mathbf{y}$. As in other generalization bounds for margin-based classifiers, we relate the generalization error to the margin of the classifier. Consider an upper bound on the above loss:

$$\mathcal{L}(\mathbf{w}, \mathbf{x}, \mathbf{y}) \leq \overline{\mathcal{L}}(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \max_{\mathbf{y}':\ \mathbf{w}^\top \mathbf{f}(\mathbf{y}) \leq \mathbf{w}^\top \mathbf{f}(\mathbf{y}')} \frac{1}{L}\ell^H(\mathbf{y}, \mathbf{y}').$$

This upper bound is tight if $\mathbf{y} = \arg\max_{\mathbf{y}'} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}')$, Otherwise, it is adversarial: it picks from all $\mathbf{y}'$ which are better ($\mathbf{w}^\top \mathbf{f}(\mathbf{y}) \leq \mathbf{w}^\top \mathbf{f}(\mathbf{y}')$), one that maximizes the Hamming distance from $\mathbf{y}$. We can now define a γ-*margin per-label loss*:

$$\mathcal{L}(\mathbf{w}, \mathbf{x}, \mathbf{y}) \leq \overline{\mathcal{L}}(\mathbf{w}, \mathbf{x}, \mathbf{y}) \leq \mathcal{L}^\gamma(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \max_{\mathbf{y}':\ \mathbf{w}^\top \mathbf{f}(\mathbf{y}) \leq \mathbf{w}^\top \mathbf{f}(\mathbf{y}') + \gamma \ell^H(\mathbf{y}, \mathbf{y}')} \frac{1}{L}\ell^H(\mathbf{y}, \mathbf{y}').$$

This upper bound is even more adversarial: it is tight if $\mathbf{y} = \arg\max_{\mathbf{y}'}[\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}') + \ell^H(\mathbf{y}, \mathbf{y}')]$, otherwise, it picks from all $\mathbf{y}'$ which are better *when helped by* $\gamma\ell^H(\mathbf{y}, \mathbf{y}')$, one that maximizes the Hamming distance from $\mathbf{y}$. Note that the loss we minimize in the max-margin formulation is very closely related (although not identical to) this upper bound.

We can now prove that the generalization accuracy of any hypothesis $\mathbf{w}$ is bounded by its empirical γ-margin per-label loss, plus a term that grows inversely with the margin.To state the bound, we need to define several other factors it depends upon. Let N_c be the maximum number of cliques in $\mathcal{G}(\mathbf{x})$, V_c be the maximum number of values in a clique $|\mathbf{Y}_c|$, q be the maximum number of cliques that have a variable in common, and R_c be an upper-bound on the 2-norm of clique basis functions. Consider a first-order sequence

model as an example, with L as the maximum length, and V the number of values a variable takes. Then $N_c = 2L - 1$ since we have L node cliques and $L - 1$ edge cliques; $V_c = V^2$ because of the edge cliques; and $q = 3$ since nodes in the middle of the sequence participate in 3 cliques: previous-current edge clique, node clique, and current-next edge clique.

Theorem 5.5.1 *For the family of hypotheses parameterized by* $\mathbf{w}$*, and any* $\delta > 0$*, there exists a constant* K *such that for any* $\gamma > 0$ *per-label margin, and* $m > 1$ *samples, the expected per-label loss is bounded by:*

$$\mathbf{E}_D[\mathcal{L}(\mathbf{w}, \mathbf{x}, \mathbf{y})] \quad \leq \quad \mathbf{E}_S[\mathcal{L}^\gamma(\mathbf{w}, \mathbf{x}, \mathbf{y})] + \sqrt{\frac{K}{m}\left[\frac{R_c^2||\mathbf{w}||^2 q^2}{\gamma^2}\left[\ln m + \ln N_c + \ln V_c\right] + \ln\frac{1}{\delta}\right]} \ ,$$

with probability at least $1 - \delta$*.* ∎

Proof: See Appendix A.1 for the proof details and the exact value of the constant K. ∎

The first term upper bounds the training error of $\mathbf{w}$. Low loss $\mathbf{E}_S[\mathcal{L}^\gamma(\mathbf{w}, \mathbf{x}, \mathbf{y})]$ at high margin γ quantifies the confidence of the prediction model. The second term depends on $||\mathbf{w}||/\gamma$, which corresponds to the complexity of the classifier (normalized by the margin level). Thus, the result provides a bound to the generalization error that trades off the *effective* complexity of the hypothesis space with the training error.

The proof uses a covering number argument analogous to previous results in SVMs [Zhang, 2002]. However we propose a novel method for covering the space of structured prediction models by using a cover of the individual clique basis function differences $\Delta\mathbf{f}_{i,c}(\mathbf{y}_c)$. This new type of cover is polynomial in the number of cliques, yielding significant improvements in the bound. Specifically, our bound has a logarithmic dependence on the number of cliques ($\ln N_c$) and depends only on the 2-norm of the basis functions per-clique (R_c). This is a significant gain over the previous result of Collins [2001] for $0/1$ loss, which has linear dependence (inside the square root) on the number of nodes (L), and depends on the joint 2-norm of all of the basis functions for an example (which is $\sim N_c R_c$). Such a result was, until now, an open problem for margin-based sequence classification [Collins, 2001]. Finally, for sequences, note that if $\frac{L}{m} = \mathcal{O}(1)$ (for example, in OCR, if the number of instances is at least a constant times the length of a word), then our bound is independent of the number of labels L.

5.6 Related work

The application of margin-based estimation methods to parsing and sequence modeling was pioneered by Collins [2001] using the Voted-Perceptron algorithm [Freund & Schapire, 1998]. He provides generalization guarantees (for $0/1$ loss) that hold for separable case and depend on the number of mistakes the perceptron makes before convergence. Remarkably, the bound does not explicitly depend on the length of the sequence, although undoubtedly the number of mistakes does.

Collins [2004] also suggested an SVM-like formulation (with exponentially many constraints) and a constraint generation method for solving it. His generalization bound (for $0/1$ loss) based on the SVM-like margin, however, has linear dependence (inside the square root) on the number of nodes (L). It also depends on the joint 2-norm of all of the basis functions for an example (which is $\sim N_c R_c$). By considering the more natural Hamming loss, we achieve a much tighter analysis.

Altun *et al.* [2003] have applied the exponential-size formulation with constraint generation we described in Sec. 4.2.1 to problems natural language processing. In a follow-up paper, Tsochantaridis *et al.* [2004] show that only a polynomial number of constraints are needed to be generated to guarantee a fixed level of precision of the solution. However, the number of constraints in many important cases is several orders higher (in L) than in the the approach we present. In addition, the corresponding problem needs to be resolved (or at least approximately resolved) after each additional constraint is added, which is prohibitively expensive for large number of examples and label variables.

The work of Guestrin *et al.* [2003] presents LP decompositions based on graphical model structure for the value function approximation problem in factored MDPs (Markov decision processes with structure). Describing the exact setting is beyond our scope, but it suffices to say that our original decomposition of the max-margin QP was inspired by the proposed technique to transform an exponential set of constraints into a polynomial one using a triangulated graph.

There has been a recent explosion of work in maximum conditional likelihood estimation of Markov networks. The work of Lafferty *et al.* [2001] has inspired many applications in natural language, computational biology, computer vision and relational modeling [Sha

& Pereira, 2003; Pinto *et al.*, 2003; Kumar & Hebert, 2003; Sutton *et al.*, 2004; Taskar *et al.*, 2002; Taskar *et al.*, 2003b]. As in the case of logistic regression, maximum conditional likelihood estimation for Markov networks can also be kernelized [Altun *et al.*, 2004; Lafferty *et al.*, 2004]. However, the solutions are non-sparse and the proposed algorithms are forced to use greedy selection of support vectors or heuristic pruning methods.

5.7 Conclusion

We use graph decomposition to derive an exact, compact, convex max-margin formulation for Markov networks with sequence and other low-treewidth structure. Our formulation avoids the exponential blow-up in the number of constraints in the max-margin QP that plagued previous approaches. The seamless integration of kernels with graphical models allows us to create very rich models that leverage the immense amount of research in kernel design and graphical model decompositions. We also use approximate graph decomposition to derive a compact approximate formulation for Markov networks in which inference is intractable.

We provide theoretical guarantees on the average *per-label* generalization error of our models in terms of the training set margin. Our generalization bound significantly tightens previous results of Collins [Collins, 2001] and suggests possibilities for analyzing per-label generalization properties of graphical models.

In the next chapter, we present an efficient algorithm that exploits graphical model inference and show experiments on a large handwriting recognition task that utilize the powerful representational capability of kernels.

Chapter 6

M^{3}N algorithms and experiments

Although the number of variables and constraints in the structured dual in Eq. (5.12) is polynomial in the size of the data, unfortunately, for standard QP solvers, the problem is often too large even for small training sets. Instead, we use a coordinate dual ascent method analogous to the sequential minimal optimization (SMO) used for SVMs [Platt, 1999].

We apply our M^{3}N framework and structured SMO algorithm to a handwriting recognition task. We show that our models significantly outperform other approaches by incorporating high-dimensional decision boundaries of polynomial kernels over character images while capturing correlations between consecutive characters.

6.1 Solving the M^{3}N QP

Let us begin by considering the primal and dual QPs for multi-class SVMs:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \Delta\mathbf{f}_i(y) \geq \ell(y) - \xi_i, \quad \forall i, y. \end{aligned} \qquad \begin{aligned} \max \quad & \sum_{i,y} \alpha_i(y)\ell_i(y) - \frac{1}{2}C\left|\left|\sum_{i,y}\alpha_i(y)\Delta\mathbf{f}_i(y)\right|\right|^2 \\ \text{s.t.} \quad & \sum_y \alpha_i(y) = 1, \ \forall i; \quad \alpha_i(y) \geq 0, \ \forall i, y. \end{aligned}$$

The KKT conditions [Bertsekas, 1999; Boyd & Vandenberghe, 2004] provide sufficient and necessary criteria for optimality of a dual solution α. As we describe below, these conditions have certain locality with respect to each example i, which allows us to perform

the search for optimal α by repeatedly considering one example at a time.

A feasible dual solution α and a primal solution defined by:

$$\begin{aligned} \mathbf{w} &= C \sum_{i,y} \alpha_i(y) \Delta \mathbf{f}_i(y) \\ \xi_i &= \max_y \left[\ell_i(y) - \mathbf{w}\Delta\mathbf{f}_i(y)\right] = \max_y \left[\ell_i(y) + \mathbf{w}^\top \mathbf{f}_i(y)\right] - \mathbf{w}^\top \mathbf{f}_i(y^{(i)}), \end{aligned} \tag{6.1}$$

are optimal if they satisfy the following two types of constraints:

$$\begin{aligned} \alpha_i(y) = 0 \;&\Rightarrow\; \mathbf{w}^\top \Delta\mathbf{f}_i(y) > \ell_i(y) - \xi_i; \qquad \text{(KKT1)} \\ \alpha_i(y) > 0 \;&\Rightarrow\; \mathbf{w}^\top \Delta\mathbf{f}_i(y) = \ell_i(y) - \xi_i. \qquad \text{(KKT2)}, \end{aligned}$$

We can express these conditions as

$$\begin{aligned} \alpha_i(y) = 0 \;&\Rightarrow\; \mathbf{w}^\top \mathbf{f}_i(y) + \ell_i(y) < \max_{y'} \left[\mathbf{w}^\top \mathbf{f}_i(y') + \ell_i(y')\right]; \qquad \text{(KKT1)} \\ \alpha_i(y) > 0 \;&\Rightarrow\; \mathbf{w}^\top \mathbf{f}_i(y) + \ell_i(y) = \max_{y'} \left[\mathbf{w}^\top \mathbf{f}_i(y') + \ell_i(y')\right]. \qquad \text{(KKT2)} \end{aligned}$$

To simplify the notation, we define

$$v_i(y) = \mathbf{w}^\top \mathbf{f}_i(y) + \ell_i(y); \quad v_i(\overline{y}) = \max_{y' \neq y} \left[\mathbf{w}^\top \mathbf{f}_i(y') + \ell_i(y')\right].$$

With these definitions, we have

$$\alpha_i(y) = 0 \Rightarrow v_i(y) < v_i(\overline{y}); \quad \text{(KKT1)} \qquad \alpha_i(y) > 0 \Rightarrow v_i(y) \geq v_i(\overline{y}); \quad \text{(KKT2)}.$$

In practice, however, we will enforce KKT conditions up to a given tolerance $0 < \epsilon \ll 1$.

$$\alpha_i(y) = 0 \Rightarrow v_i(y) \leq v_i(\overline{y}) + \epsilon; \qquad \alpha_i(y) > 0 \Rightarrow v_i(y) \geq v_i(\overline{y}) - \epsilon. \tag{6.2}$$

Essentially, $\alpha_i(y)$ can be *zero* only if $v_i(y)$ is at most ϵ *larger than the all others*. Conversely, $\alpha_i(y)$ can be *non-zero* only if $v_i(y)$ is at most ϵ *smaller than the all others*.

Note that the normalization constraints on the dual variables α are local to each example i. This allows us to perform dual block-coordinate ascent where a block corresponds to

1. Initialize: $\alpha_i(y) = \mathbb{1}(y = y^{(i)}), \quad \forall\, i, y.$
2. Set $violation = 0$,
3. For each i,
4. If α_i violates (KKT1) or (KKT2),
5. Set $violation = 1$,
6. Find feasible α_i' such that $\mathcal{Q}(\alpha_i', \alpha_{-i}) > \mathcal{Q}(\alpha_i, \alpha_{-i})$ and set $\alpha_i = \alpha_i'$.
7. If $violation = 1$ goto 2.

Figure 6.1: Block-coordinate dual ascent.

the vector of dual variables α_i for a single example i. The general form of block-coordinate ascent algorithm as shown in Fig. 6.1 is essentially coordinate ascent on blocks α_i, maintaining the feasibility of the dual. When optimizing with respect to a single block i, the objective function can be split into two terms:

$$\mathcal{Q}(\alpha) = \mathcal{Q}(\alpha_{-i}) + \mathcal{Q}(\alpha_i, \alpha_{-i}),$$

where α_{-i} denotes all dual α_k variables for k other than i. Only the second part of the objective $\mathcal{Q}(\alpha_i, \alpha_{-i})$ matters for optimizing with respect to α_i. The algorithm starts with a feasible dual solution α and improves the objective block-wise until all KKT conditions are satisfied. Checking the constraints requires computing $\mathbf{w}$ and ξ from α according to Eq. (6.1).

As long as the local ascent step over α_i is guaranteed to improve the objective when KKT conditions are violated, the algorithm will converge to the global maximum in a finite number of steps (within the precision). This allows us to focus on efficient updates to a single block of α_i at a time.

Let $\alpha_i'(y) = \alpha_i(y) + \lambda(y)$. Note that $\sum_y \lambda(y) = 0$ and $\alpha_i(y) + \lambda(y) \geq 0$ so that α_i' is feasible. We can write the objective $\mathcal{Q}(\alpha_{-i}) + \mathcal{Q}(\alpha_i', \alpha_{-i})$ in terms of λ and α:

$$\sum_{j,y} \alpha_j(y)\ell_j(y) + \sum_y \lambda(y)\ell_i(y) - \frac{1}{2}C \left|\left| \sum_y \lambda(y)\Delta\mathbf{f}_i(y) + \sum_{j,y} \alpha_j(y)\Delta\mathbf{f}_j(y) \right|\right|^2 .$$

By dropping all terms that do not involve λ, and making the substitution $\mathbf{w} = C\sum_{j,y}\alpha_j(y)\Delta\mathbf{f}_j(y)$, we get:

$$\sum_y \lambda(y)\ell_i(y) - \mathbf{w}^\top\left(\sum_y \lambda(y)\Delta\mathbf{f}_i(y)\right) - \frac{1}{2}C\left|\left|\sum_y \lambda(y)\Delta\mathbf{f}_i(y)\right|\right|^2.$$

Since $\sum_y \lambda(y) = 0$,

$$\sum_y \lambda(y)\Delta\mathbf{f}_i(y) = \sum_y \lambda(y)\mathbf{f}_i(y^{(i)}) - \sum_y \lambda(y)\mathbf{f}_i(y) = -\sum_y \lambda(y)\mathbf{f}_i(y).$$

Below we also make the substitution $v_i(y) = \mathbf{w}^\top\mathbf{f}_i(y) + \ell_i(y)$ to get the optimization problem for λ:

$$\begin{aligned} \max \quad & \sum_y \lambda(y)v_i(y) - \frac{1}{2}C\left|\left|\sum_y \lambda(y)\mathbf{f}_i(y)\right|\right|^2 \\ \text{s.t.} \quad & \sum_y \lambda(y) = 0; \quad \alpha_i(y) + \lambda(y) \geq 0, \ \forall y. \end{aligned}$$

6.1.1 SMO

We do not need to solve the optimization subproblem above at each pass through the data. All that is required is an ascent step, not a full optimization. Sequential Minimal Optimization (SMO) approach takes an ascent step that modifies the least number of variables. In our case, we have the simplex constraint, so we must change at least two variables in order to respect the normalization constraint (by moving weight from one dual variable to another). We address a strategy for selecting the two variables in the next section, but for now assume we have picked $\lambda(y')$ and $\lambda(y'')$. Then we have $\delta = \lambda(y') = -\lambda(y'')$ in order to sum to 1. The optimization problem becomes a single variable quadratic program in δ:

$$\begin{aligned} \max \quad & [v_i(y') - v_i(y'')]\delta - \frac{1}{2}C||\mathbf{f}_i(y') - \mathbf{f}_i(y'')||^2\delta^2 \\ \text{s.t.} \quad & \alpha_i(y') + \delta \geq 0; \quad \alpha_i(y'') - \delta \geq 0. \end{aligned} \tag{6.3}$$

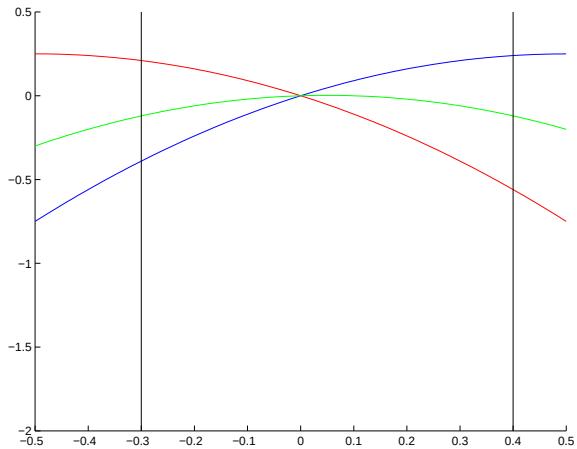


Figure 6.2: Representative examples of the SMO subproblem. Horizonal axis represents δ with two vertical lines depicting the upper and lower bounds c and d. Vertical axis represents the objective. Optimum either occurs at the maximum of the parabola if it is feasible or the upper or lower bound otherwise.

With $a = v_i(y') - v_i(y'')$, $b = C||\mathbf{f}_i(y') - \mathbf{f}_i(y'')||^2$, $c = -\alpha_i(y')$, $d = \alpha_i(y'')$, we have:

$$\max\,[a\delta - \frac{b}{2}\delta^2] \;\text{ s.t. }\; c \leq \delta \leq d, \tag{6.4}$$

where the optimum is achieved at the maximum of the parabola a/b if $c \leq a/b \leq d$ or at the boundary c or d (see Fig. 6.1.1). Hence the solution is given by simply clipping a/b:

$$\delta^* = \max(c, \min(d, a/b)).$$

The key advantage of SMO is the simplicity of this update. Computing the coefficients involves dot products (or kernel evaluations) to compute $\mathbf{w}^\top \mathbf{f}_i(y')$ and $\mathbf{w}^\top \mathbf{f}_i(y'')$ as well as $(\mathbf{f}_i(y') - \mathbf{f}_i(y''))^\top(\mathbf{f}_i(y') - \mathbf{f}_i(y''))$.

6.1.2 Selecting SMO pairs

How do we actually select such a pair to guarantee that we make progress in optimizing the objective? Note that at least one of the assignments y must violate (KKT1) or (KKT2),

1. Set $violation = 0$.
2. For each y,
3. KKT1: If $\alpha_i(y) = 0$ and $v_i(y) > v_i(\overline{y}) + \epsilon$,
4. Set $y' = y$ and $violation = 1$ and goto 7.
5. KKT2: If $\alpha_i(y) > 0$ and $v_i(y) < v_i(\overline{y}) - \epsilon$,
6. Set $y' = y$ and $violation = 2$ and goto 7.
7. If $violation > 0$,
8. For each $y \neq y'$,
9. If $violation = 1$ and $\alpha_i(y) > 0$,
10. Set $y'' = y$ and goto 13.
11. If $violation = 2$ and $v_i(y) > v_i(y')$,
12. Set $y'' = y$ and goto 13.
13. Return y' and y''.

Figure 6.3: SMO pair selection.

because otherwise α_i is optimal with respect to the current α_{-i}. The selection algorithm is outlined in Fig. 6.3.

The first variable in the pair, y', corresponds to a violated condition, while the second variable, y'', is chosen to guarantee that solving Eq. (6.3) will result in improving the objective. There are two cases, corresponding to violation of KKT1 and violation of KKT2.

Case KKT1. $\alpha_i(y') = 0$ but $v_i(y') > v_i(\overline{y'}) + \epsilon$. This is the case where i, y' is a not support vector but should be. We would like to increase $\alpha_i(y')$, so we need $\alpha_i(y'') > 0$ to borrow from. There will always be a such a y'' since $\sum_y \alpha_i(y) = 1$ and $\alpha_i(y') = 0$. Since $v_i(y') > v_i(\overline{y'}) + \epsilon$, $v_i(y') > v_i(y'') + \epsilon$, so the linear coefficient in Eq. (6.4) is $a = v_i(y') - v_i(y'') > \epsilon$. Hence the unconstrained maximum is positive $a/b > 0$. Since the upper-bound $d = \alpha_i(y'') > 0$, we have enough freedom to improve the objective.

Case KKT2. $\alpha_i(y') > 0$ but $v_i(y') < v_i(\overline{y'}) - \epsilon$. This is the case where i, y' is a support vector but should not be. We would like to decrease $\alpha_i(y')$, so we need $v_i(y'') > v_i(y')$ so that $a/b < 0$. There will always be a such a y'' since $v_i(y') < v_i(\overline{y'}) - \epsilon$. Since the

lower-bound $c = -\alpha_i(y') < 0$, again we have enough freedom to improve the objective.

Since at each iteration we are guaranteed to improve the objective if the KKT conditions are violated and the objective is bounded, we can use the SMO in the block-coordinate ascent algorithm to converge in a finite number of steps. To the best of our knowledge, there are no upper bounds on the speed of convergence of SMO, but experimental evidence has shown it a very effective algorithm for SVMs [Platt, 1999]. Of course, we can improve the speed of convergence by adding heuristics in the selection of the pair, as long as we guarantee that improvement is possible when KKT conditions are violated.

6.1.3 Structured SMO

Clearly, we cannot perform the above SMO updates in the space of α directly for the structured problems, since the number of α variables is exponential. The constraints on μ variables are much more complicated, since each μ participates not only in non-negativity and normalization constraints, but also clique-agreement constraints. We cannot limit our ascent steps to changing only two μ variables at a time, because in order to make a change in one clique and stay feasible, we need to modify variables in overlapping cliques. Fortunately, we can perform SMO updates on α variables implicitly in terms of the marginal dual variables μ.

The diagram in Fig. 6.1.3 shows the abstract outline of the algorithm. The key steps in the SMO algorithm are checking for violations of the KKT conditions, selecting the pair $\mathbf{y}'$ and $\mathbf{y}''$, computing the corresponding coefficients a, b, c, d and updating the dual. We will show how to do these operations by doing all the hard work in terms of the polynomially many marginal μ_i variables and auxiliary "max-marginals" variables.

Structured KKT conditions

As before, we define $v_i(\mathbf{y}) = \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$. The KKT conditions are, for all $\mathbf{y}$:

$$\alpha_i(\mathbf{y}) = 0 \Rightarrow v_i(\mathbf{y}) \leq v_i(\overline{\mathbf{y}}); \qquad \alpha_i(\mathbf{y}) > 0 \Rightarrow v_i(\mathbf{y}) \geq v_i(\overline{\mathbf{y}}). \tag{6.5}$$

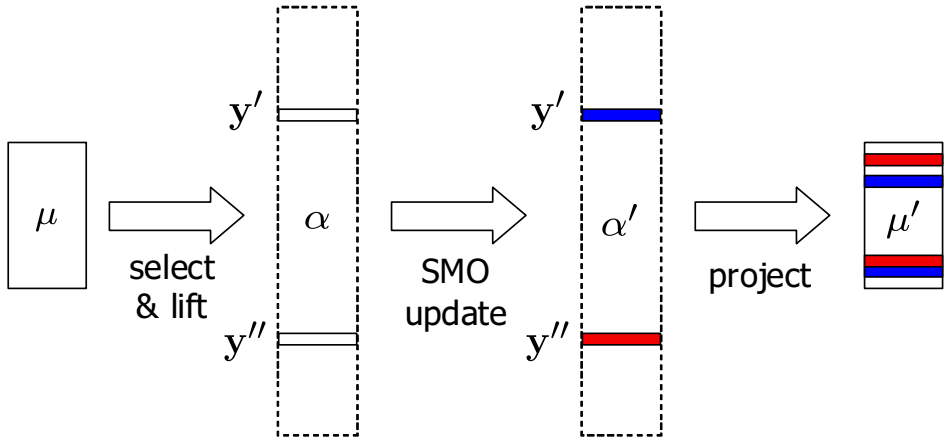


Figure 6.4: Structured SMO diagram. We use marginals μ to select an appropriate pair of instantiations $\mathbf{y}'$ and $\mathbf{y}''$ and reconstruct their α values. We then perform the simple SMO update and project the result back onto the marginals.

Of course, we cannot check these explicitly. Instead, we define max-marginals for each clique in the junction tree $c \in \mathcal{T}^{(i)}$ and its values $\mathbf{y}_c$, as:

$$\widehat{v}_{i,c}(\mathbf{y}_c) = \max_{\mathbf{y}\sim\mathbf{y}_c} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})], \qquad \widehat{\alpha}_{i,c}(\mathbf{y}_c) = \max_{\mathbf{y}\sim\mathbf{y}_c} \alpha_i(\mathbf{y}).$$

We also define $\widehat{v}_{i,c}(\overline{\mathbf{y}_c}) = \max_{\mathbf{y}'_c \neq \mathbf{y}_c} \widehat{v}_{i,c}(\mathbf{y}'_c) = \max_{\mathbf{y}\not\sim\mathbf{y}_c} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$. Note that we do not explicitly represent $\alpha_i(\mathbf{y})$, but we can reconstruct the maximum-entropy one from the marginals μ_i by using Eq. (5.13). Both $\widehat{v}_{i,c}(\mathbf{y}_c)$ and $\widehat{\alpha}_{i,c}(\mathbf{y}_c)$ can be computed by using the Viterbi algorithm (one pass propagation towards the root and one outwards from the root [Cowell *et al.*, 1999]). We can now express the KKT conditions in terms of the max-marginals for each clique $c \in \mathcal{T}^{(i)}$ and its values $\mathbf{y}_c$:

$$\widehat{\alpha}_{i,c}(\mathbf{y}_c) = 0 \Rightarrow \widehat{v}_{i,c}(\mathbf{y}_c) \leq \widehat{v}_{i,c}(\overline{\mathbf{y}_c}); \qquad \widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0 \Rightarrow \widehat{v}_{i,c}(\mathbf{y}_c) \geq \widehat{v}_{i,c}(\overline{\mathbf{y}_c}). \tag{6.6}$$

Theorem 6.1.1 *The KKT conditions in Eq. (6.5) and Eq. (6.6) are equivalent.*

Proof:

Eq. (6.5) $\Rightarrow$ Eq. (6.6). Assume Eq. (6.5). Suppose, we have a violation of KKT1: for some $c, \mathbf{y}_c$, $\widehat{\alpha}_{i,c}(\mathbf{y}_c) = 0$, but $\widehat{v}_{i,c}(\mathbf{y}_c) > \widehat{v}_{i,c}(\overline{\mathbf{y}_c})$. Since $\widehat{\alpha}_{i,c}(\mathbf{y}_c) = \max_{\mathbf{y}\sim\mathbf{y}_c} \alpha_i(\mathbf{y}) = 0$,

then $\alpha_i(\mathbf{y}) = 0,\ \forall \mathbf{y} \sim \mathbf{y}_c$. Hence, by Eq. (6.5), $v_i(\mathbf{y}) \leq v_i(\overline{\mathbf{y}}),\ \forall \mathbf{y} \sim \mathbf{y}_c$. But $\widehat{v}_{i,c}(\mathbf{y}_c) > \widehat{v}_{i,c}(\overline{\mathbf{y}_c})$ implies the opposite: there exists $\mathbf{y} \sim \mathbf{y}_c$ such that $v_i(\mathbf{y}) > \widehat{v}_{i,c}(\overline{\mathbf{y}_c})$, which also implies $v_i(\mathbf{y}) > v_i(\overline{\mathbf{y}})$, a contradiction.

Now suppose we have a violation of KKT2: for some $i, \mathbf{y}_c$, $\widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0$, but $\widehat{v}_{i,c}(\mathbf{y}_c) < \widehat{v}_{i,c}(\overline{\mathbf{y}_c})$. Then $v_i(\mathbf{y}) < v_i(\overline{\mathbf{y}}),\ \forall \mathbf{y} \sim \mathbf{y}_c$. But $\widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0$ implies there exists $\mathbf{y} \sim \mathbf{y}_c$ such that $\alpha_i(\mathbf{y}) > 0$. For that $\mathbf{y}$, by Eq. (6.5), $v_i(\mathbf{y}) \geq v_i(\overline{\mathbf{y}})$, a contradiction.

Eq. (6.6) $\Rightarrow$ Eq. (6.5). Assume Eq. (6.6). Suppose we have a violation of KKT1: for some $\mathbf{y}$, $\alpha_i(\mathbf{y}) = 0$, but $v_i(\mathbf{y}) > v_i(\overline{\mathbf{y}})$. This means that $\mathbf{y}$ is the optimum of $v_i(\cdot)$, hence $\widehat{v}_{i,c}(\mathbf{y}_c) = v_i(\mathbf{y}) > v_i(\overline{\mathbf{y}}) > \widehat{v}_{i,c}(\overline{\mathbf{y}_c}),\ \forall c \in \mathcal{T}^{(i)}, \mathbf{y}_c \sim \mathbf{y}$. But by Eq. (6.6), if $\widehat{v}_{i,c}(\mathbf{y}_c) > \widehat{v}_{i,c}(\overline{\mathbf{y}_c})$, then we cannot have $\widehat{\alpha}_{i,c}(\mathbf{y}_c) = 0$. Hence all the $\mathbf{y}$-consistent α_i max-marginals are positive $\widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0,\ \forall c \in \mathcal{T}^{(i)}$, and it follows that all the $\mathbf{y}$-consistent marginals μ_i are positive as well $\mu_{i,c}(\mathbf{y}_c) > 0,\ \forall c \in \mathcal{T}^{(i)}$ (since sum upper-bounds max). But $\alpha_i(\mathbf{y}) = \frac{\prod_{c \in \mathcal{T}^{(i)}} \mu_{i,c}(\mathbf{y}_c)}{\prod_{c \in \mathcal{S}^{(i)}} \mu_{i,s}(\mathbf{y}_s)}$, so if all the $\mathbf{y}$-consistent marginals are positive, then $\alpha_i(\mathbf{y}) > 0$, a contradiction.

Now suppose we have a violation of KKT2: for some $\mathbf{y}$, $\alpha_i(\mathbf{y}) > 0$, but $v_i(\mathbf{y}) < v_i(\overline{\mathbf{y}})$. Since $\alpha_i(\mathbf{y}) > 0$, we know that all the $\mathbf{y}$-consistent α_i max-marginals are positive $\widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0,\ \forall c \in \mathcal{T}^{(i)}$. By Eq. (6.6), $\widehat{v}_{i,c}(\mathbf{y}_c) \geq \widehat{v}_{i,c}(\overline{\mathbf{y}_c}),\ \forall c \in \mathcal{T}^{(i)}$. Note that trivially $\max_{\mathbf{y}'} v_i(\mathbf{y}') = \max(\widehat{v}_{i,c}(\mathbf{y}'_c), \widehat{v}_{i,c}(\overline{\mathbf{y}'_c}))$ for any clique c and clique assignment $\mathbf{y}'_c$. Since $\widehat{v}_{i,c}(\mathbf{y}_c) \geq \widehat{v}_{i,c}(\overline{\mathbf{y}_c}),\ \forall c \in \mathcal{T}^{(i)}$, then $\max_{\mathbf{y}'} v_i(\mathbf{y}') = \widehat{v}_{i,c}(\mathbf{y}_c), ,\ \forall c \in \mathcal{T}^{(i)}$. That is, $\widehat{v}_{i,c}(\mathbf{y}_c)$ is the optimal value. We will show that $v_i(\mathbf{y}) = \widehat{v}_{i,c}(\mathbf{y}_c)$, a contradiction. To show that this, we consider any two adjacent nodes in the tree $\mathcal{T}^{(i)}$, cliques a and b, with a separator s, and show that $\widehat{v}_{i,a\cup b}(\mathbf{y}_{a\cup b}) = \widehat{v}_{i,a}(\mathbf{y}_a) = \widehat{v}_{i,b}(\mathbf{y}_b)$. By chaining this equality from the root of the tree to all the leaves, we get $v_i(\mathbf{y}) = \widehat{v}_{i,c}(\mathbf{y}_c)$ for any c.

We need to introduce some more notation to deal with the two parts of the tree induced by cutting the edge between a and b. Let $\{A, B\}$ be a partition of the nodes $\mathcal{T}^{(i)}$ (cliques of $\mathcal{C}^{(i)}$) resulting from removing the edge between a and b such that $a \in A$ and $b \in B$. We denote the two subsets of an assignment $\mathbf{y}$ as $\mathbf{y}_A$ and $\mathbf{y}_B$ (with overlap at $\mathbf{y}_s$). The value of an assignment $v_i(\mathbf{y})$ can be decomposed into two parts: $v_i(\mathbf{y}) = v_{i,A}(\mathbf{y}_A) + v_{i,B}(\mathbf{y}_B)$, where $v_{i,A}(\mathbf{y}_A)$ and $v_{i,B}(\mathbf{y}_B)$ only count the contributions of their constituent cliques. Take any maximizer, $\mathbf{y}^{(a)} \sim \mathbf{y}_a$ with $v_i(\mathbf{y}^{(a)}) = \widehat{v}_{i,a}(\mathbf{y}_a) \geq \widehat{v}_{i,a}(\overline{\mathbf{y}_a})$ and any maximizer $\mathbf{y}^{(b)} \sim \mathbf{y}_b$ with $v_i(\mathbf{y}^{(b)}) = \widehat{v}_{i,b}(\mathbf{y}_b) \geq \widehat{v}_{i,b}(\overline{\mathbf{y}_b})$, which by definition agree with $\mathbf{y}$

on the intersection s. We decompose the two associated values into the corresponding parts: $v_i(\mathbf{y}^{(a)}) = v_i(\mathbf{y}_A^{(a)}) + v_i(\mathbf{y}_B^{(a)})$ and $v_i(\mathbf{y}^{(b)}) = v_i(\mathbf{y}_A^{(b)}) + v_i(\mathbf{y}_B^{(b)})$. We create a new assignment that combines the best of the two: $\mathbf{y}^{(s)} = \mathbf{y}_A^{(b)} \cup \mathbf{y}_B^{(a)}$. Note that $v_i(\mathbf{y}^{(s)}) = v_i(\mathbf{y}_A^{(b)}) + v_i(\mathbf{y}_B^{(a)}) = \widehat{v}_{i,s}(\mathbf{y}_s)$, since we essentially fixed the intersection s and maximized over the rest of the variables in A and B separately. Now $\widehat{v}_{i,a}(\mathbf{y}_a) = \widehat{v}_{i,b}(\mathbf{y}_b) \geq \widehat{v}_{i,s}(\mathbf{y}_s)$ since they are optimal as we said above. Hence we have $v_i(\mathbf{y}_A^{(a)}) + v_i(\mathbf{y}_B^{(a)}) = v_i(\mathbf{y}_A^{(b)}) + v_i(\mathbf{y}_B^{(b)}) \geq v_i(\mathbf{y}_A^{(b)}) + v_i(\mathbf{y}_B^{(a)})$ which implies that $v_i(\mathbf{y}_A^{(a)}) \geq v_i(\mathbf{y}_A^{(b)})$ and $v_i(\mathbf{y}_B^{(b)}) \geq v_i(\mathbf{y}_B^{(a)})$. Now we create another assignment that clamps the value of both a and b: $\mathbf{y}^{(a \cup b)} = \mathbf{y}_A^{(a)} \cup \mathbf{y}_B^{(b)}$. The value of this assignment is optimal $v_i(\mathbf{y}^{(a \cup b)}) = v_i(\mathbf{y}_A^{(a)}) + v_i(\mathbf{y}_B^{(b)}) = v_i(\mathbf{y}^{(a)}) = v_i(\mathbf{y}^{(b)})$. ∎

Structured SMO pair selection and update

As in multi-class problems, we will select the first variable in the pair, $\mathbf{y}'$, corresponding to a violated condition, while the second variable, $\mathbf{y}''$, to guarantee that solving Eq. (6.3) will result in improving the objective. Having selected $\mathbf{y}'$ and $\mathbf{y}''$, the coefficients for the one-variable QP in Eq. (6.4) are $a = v_i(\mathbf{y}') - v_i(\mathbf{y}'')$, $b = C||\mathbf{f}_i(\mathbf{y}') - \mathbf{f}_i(\mathbf{y}'')||^2$, $c = -\alpha_i(\mathbf{y}')$, $d = \alpha_i(\mathbf{y}'')$. As before, we enforce approximate KKT conditions in the algorithm in Fig. 6.5. We have two cases, corresponding to violation of KKT1 and violation of KKT2.

Case KKT1. $\widehat{\alpha}_{i,c}(\mathbf{y}'_c) = 0$ but $\widehat{v}_{i,c}(\mathbf{y}'_c) > \widehat{v}_{i,c}(\overline{\mathbf{y}'_c}) + \epsilon$. We have set $\mathbf{y}' = \arg\max_{\mathbf{y} \sim \mathbf{y}_c} v_i(\mathbf{y})$, so $v_i(\mathbf{y}') = \widehat{v}_{i,c}(\mathbf{y}'_c) > \widehat{v}_{i,c}(\overline{\mathbf{y}'_c}) + \epsilon > v_i(\overline{\mathbf{y}'}) + \epsilon$ and $\alpha_i(\mathbf{y}') = 0$. This is the case where $i, \mathbf{y}'$ is a not support vector but should be. We would like to increase $\alpha_i(\mathbf{y}')$, so we need $\alpha_i(\mathbf{y}'') > 0$ to borrow from. There will always be a such a y'' (with $\mathbf{y}''_c \neq \mathbf{y}'_c$) since $\sum_y \alpha_i(\mathbf{y}) = 1$ and $\alpha_i(\mathbf{y}') = 0$. We can find one by choosing $\mathbf{y}_c$ for which $\widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0$, which guarantees that for $\mathbf{y}''_c = \arg\max_{\mathbf{y} \sim \mathbf{y}_c} \alpha_i(\mathbf{y})$, $\alpha_i(\mathbf{y}'') > 0$. Since $v_i(\mathbf{y}') \geq v_i(\overline{\mathbf{y}'}) + \epsilon$, $v_i(\mathbf{y}') \geq v_i(\mathbf{y}'') + \epsilon$, so the linear coefficient in Eq. (6.4) is $a = v_i(\mathbf{y}') - v_i(\mathbf{y}'') > \epsilon$. Hence the unconstrained maximum is positive $a/b > 0$. Since the upper-bound $d = \alpha_i(\mathbf{y}'') > 0$, we have enough freedom to improve the objective.

Case KKT2. $\widehat{\alpha}_{i,c}(\mathbf{y}'_c) > 0$ but $\widehat{v}_{i,c}(\mathbf{y}'_c) < \widehat{v}_{i,c}(\overline{\mathbf{y}'_c}) - \epsilon$. We have set $\mathbf{y}' = \arg\max_{\mathbf{y} \sim \mathbf{y}_c} \alpha_i(\mathbf{y})$, so $\alpha_i(\mathbf{y}') = \widehat{\alpha}_{i,c}(\mathbf{y}'_c) > 0$ and $v_i(\mathbf{y}') < \widehat{v}_{i,c}(\mathbf{y}'_c) < \widehat{v}_{i,c}(\overline{\mathbf{y}'_c}) - \epsilon < v_i(\overline{\mathbf{y}'}) - \epsilon$. This is the case where $i, \mathbf{y}'$ is a support vector but should not be. We would like to decrease $\alpha_i(\mathbf{y}')$, so we need $v_i(\mathbf{y}'') > v_i(\mathbf{y}')$ so that $a/b < 0$. There will always be a such a $\mathbf{y}''$ since

1. Set $violation = 0$.
2. For each $c \in \mathcal{T}^{(i)}$, $\mathbf{y}_c$
3. KKT1: If $\widehat{\alpha}_{i,c}(\mathbf{y}_c) = 0$, and $\widehat{v}_{i,c}(\mathbf{y}_c) > \widehat{v}_{i,c}(\overline{\mathbf{y}_c}) + \epsilon$,
4. Set $\mathbf{y}'_c = \mathbf{y}_c$, $\mathbf{y}' = \arg\max_{\mathbf{y}\sim\mathbf{y}_c} v_i(\mathbf{y})$ and $violation = 1$ and goto 7.
5. KKT2: If $\widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0$, and $\widehat{v}_{i,c}(\mathbf{y}_c) < \widehat{v}_{i,c}(\overline{\mathbf{y}_c}) - \epsilon$,
6. Set $\mathbf{y}'_c = \mathbf{y}_c$, $\mathbf{y}' = \arg\max_{\mathbf{y}\sim\mathbf{y}_c} \alpha_i(\mathbf{y})$ and $violation = 2$ and goto 7.
7. If $violation > 0$,
8. For each $\mathbf{y}_c \neq \mathbf{y}'_c$,
9. If $violation = 1$ and $\widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0$,
10. Set $\mathbf{y}''_c = \arg\max_{\mathbf{y}\sim\mathbf{y}_c} \alpha_i(\mathbf{y})$ and goto 13.
11. If $violation = 2$ and $\widehat{v}_{i,c}(\mathbf{y}_c) > \widehat{v}_{i,c}(\mathbf{y}'_c)$,
12. Set $\mathbf{y}'' = \arg\max_{\mathbf{y}\sim\mathbf{y}_c} v_i(\mathbf{y})$ and goto 13.
13. Return $\mathbf{y}'$ and $\mathbf{y}''$.

Figure 6.5: Structured SMO pair selection.

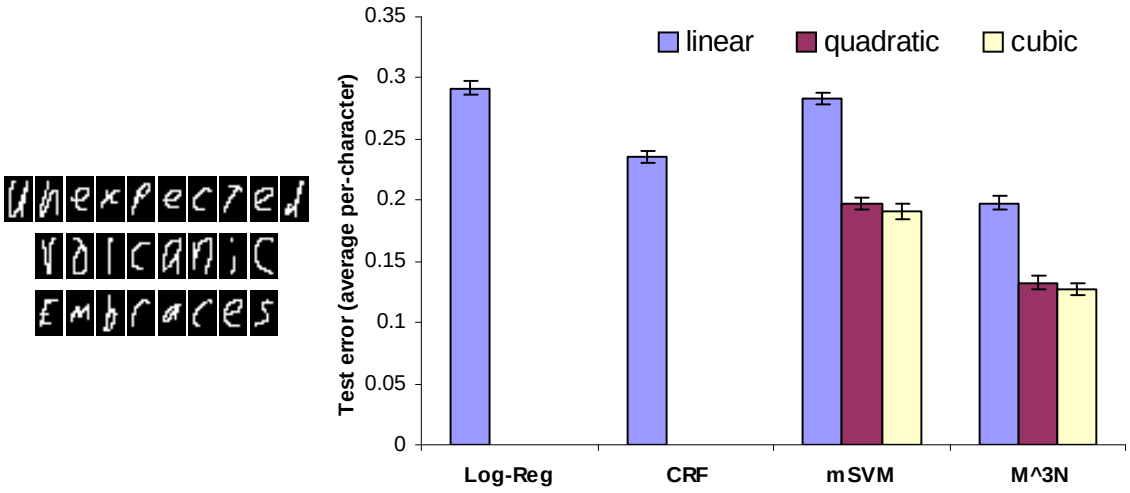
$v_i(\mathbf{y}') < v_i(\overline{y'}) - \epsilon$. We can find one by choosing $\mathbf{y}_c$ for which $\widehat{v}_{i,c}(\mathbf{y}_c) > \widehat{v}_{i,c}(\mathbf{y}_c) - \epsilon$, which guarantees that for $\mathbf{y}''_c = \arg\max_{\mathbf{y}\sim\mathbf{y}_c} v_i(\mathbf{y})$, $v_i(\mathbf{y}'') > v_i(\mathbf{y}') - \epsilon$, Since the lower-bound $c = -\alpha_i(\mathbf{y}') < 0$, again we have enough freedom to improve the objective.

Having computed new values $\alpha'_i(\mathbf{y}') = \alpha_i(\mathbf{y}') + \delta$ and $\alpha'_i(\mathbf{y}'') = \alpha_i(\mathbf{y}') - \delta$, we need to project this change onto the marginal dual variables μ_i. The only marginal affected are the ones consistent with $\mathbf{y}'$ and/or $\mathbf{y}''$, and the change is very simple:

$$\mu'_{i,c}(\mathbf{y}_c) = \mu_{i,c}(\mathbf{y}_c) + \delta\mathbb{1}(\mathbf{y}_c \sim \mathbf{y}') - \delta\mathbb{1}(\mathbf{y}_c \sim \mathbf{y}'').$$

6.2 Experiments

We selected a subset of ~ 6100 handwritten words, with average length of ~ 8 characters, from 150 human subjects, from the data set collected by Kassel [1995]. Each word was divided into characters, each character was rasterized into an image of 16 by 8 binary



(a) (b)

Figure 6.6: (a) 3 example words from the OCR data set; (b) OCR: Average per-character test error for logistic regression, CRFs, multiclass SVMs, and M^{3}Ns, using linear, quadratic, and cubic kernels.

pixels. (See Fig. 6.6(a).) In our framework, the image for each word corresponds to $\mathbf{x}$, a label of an individual character to $\mathcal{Y}_j$, and a labeling for a complete word to $\mathcal{Y}$. Each label $\mathcal{Y}_j$ takes values from one of 26 classes $\{a, \ldots, z\}$.

The data set is divided into 10 folds of ~ 600 training and ~ 5500 testing examples. The accuracy results, summarized in Fig. 6.6(b), are averages over the 10 folds. We implemented a selection of state-of-the-art classification algorithms: *independent label approaches*, which do not consider the correlation between neighboring characters — logistic regression, multi-class SVMs as described in Eq. (2.9), and one-against-all SVMs (whose performance was slightly lower than multi-class SVMs); and *sequence approaches* — CRFs, and our proposed M^3 networks. Logistic regression and CRFs are both trained by maximizing the conditional likelihood of the labels given the features, using a zero-mean diagonal Gaussian prior over the parameters, with a standard deviation between 0.1 and 1. The other methods are trained by margin maximization. Our features for each label $\mathcal{Y}_j$ are the corresponding image of ith character. For the sequence approaches (CRFs and M^3), we used an indicator basis function to represent the correlation between $\mathcal{Y}_j$ and $\mathcal{Y}_{i+1}$.

For margin-based methods (SVMs and M^3), we were able to use kernels (both quadratic and cubic were evaluated) to increase the dimensionality of the feature space. We used the structured SMO algorithm with about 30-40 iterations through the data. Using these high-dimensional feature spaces in CRFs is not feasible because of the enormous number of parameters.

Fig. 6.6(b) shows two types of gains in accuracy: First, by using kernels, margin-based methods achieve a very significant gain over the respective likelihood maximizing methods. Second, by using sequences, we obtain another significant gain in accuracy. Interestingly, the error rate of our method using linear features is 16% lower than that of CRFs, and about the same as multi-class SVMs with cubic kernels. Once we use cubic kernels our error rate is 45% lower than CRFs and about 33% lower than the best previous approach. For comparison, the previously published results, although using a different setup (e.g., a larger training set), are about comparable to those of multiclass SVMs.

6.3 Related work

The kernel-adatron [Friess *et al.*, 1998] and voted-perceptron algorithms [Freund & Schapire, 1998] for large-margin classifiers have a similar online optimization scheme. Collins [2001] have applied voted-perceptron to structured problems in natural language. Although head-to-head comparisons have not been performed, it seems that, empirically, less passes (about 30-40) are needed for our algorithm than in the perceptron literature.

Recently, the Exponentiated Gradient [Kivinen & Warmuth, 1997] algorithm has been adopted to solve our structured QP for max-margin estimation [Bartlett *et al.*, 2004]. Although the EG algorithm has attractive convergence properties, it has yet to be shown to learn faster than Structured SMO, particularly in the early iterations through the dataset.

6.4 Conclusion

In this chapter, we address the large (though polynomial) size of our quadratic program using an effective optimization procedure inspired by SMO. In our experiments with the OCR task, our sequence model significantly outperforms other approaches by incorporating

high-dimensional decision boundaries of polynomial kernels over character images while capturing correlations between consecutive characters. Overall, we believe that M^3 networks will significantly further the applicability of high accuracy margin-based methods to real-world structured data. In the next two chapters, we apply this framework to important classes of Markov networks for spatial and relational data.

Chapter 7

Associative Markov networks

In the previous chapter, we considered applications of sequence-structured Markov networks, which allow very efficient inference and learning. The chief computational bottleneck in applying Markov networks for other large-scale prediction problems is inference, which is NP-hard in general networks suitable in a broad range of practical Markov network structures, including grid-topology networks [Besag, 1986].

One can address the tractability issue by limiting the structure of the underlying network. In some cases, such as the quad-tree model used for image segmentation [Bouman & Shapiro, 1994], a tractable structure is determined in advance. In other cases (e.g., [Bach & Jordan, 2001]), the network structure is learned, subject to the constraint that inference on these networks is tractable. In many cases, however, the topology of the Markov network does not allow tractable inference. For example, in hypertext, the network structure can mirror the hyperlink graph, which is usually highly interconnected, leading to computationally intractable networks.

In this chapter, we show that optimal learning is feasible for an important subclass of Markov networks — networks with *attractive potentials*. This subclass, called *associative Markov networks (AMNs)*, contains networks of discrete variables with K labels and arbitrary-size clique potentials with K parameters that favor the same labels for all variables in the clique. Such positive interactions capture the "guilt by association" pattern of reasoning present in many domains, in which connected ("associated") variables tend to have the same label. AMNs are a natural fit object recognition and segmentation, webpage

classification, and many other applications.

In the max-margin estimation framework, the inference subtask is one of finding the best joint (MAP) assignment to all of the variables in a Markov network. By contrast, other learning tasks (e.g., maximizing the conditional likelihood of the target labels given the features) require that we compute the posterior probabilities of different label assignments, rather than just the MAP.

The MAP problem can naturally be expressed as an integer programming problem. We use a linear program relaxation of this integer program in the min-max formulation. We show that, for associative Markov networks of over binary variables ($K = 2$), this linear program provides exact answers. To our knowledge, our method is the first to allow training Markov networks of arbitrary connectivity and topology. For the non-binary case ($K > 2$), the approximate linear program is not guaranteed to be optimal but we can bound its relative error. Our empirical results suggest that the solutions of the resulting approximate max-margin formulation work well in practice.

We present an AMN-based method for object segmentation of complex from 3D range data. By constraining the class of Markov networks to AMNs, our models can be learned efficiently and at run-time, scale up to tens of millions of nodes and edges. The proposed learning formulation effectively and directly learns to exploit a large set of complex surface and volumetric features, while balancing the spatial coherence modeled by the AMN.

7.1 Associative networks

Associative interactions arise naturally in the context of image processing, where nearby pixels are likely to have the same label [Besag, 1986; Boykov *et al.*, 1999b]. In this setting, a common approach is to use a *generalized Potts model* [Potts, 1952], which penalizes assignments that do not have the same label across the edge: $\phi_{ij}(k, l) = \lambda_{ij},\ \forall k \neq l$ and $\phi_{ij}(k, k) = 1$, where $\lambda_{ij} \leq 1$.

For binary-valued Potts models, Greig *et al.* [1989] show that the MAP problem can be formulated as a min-cut in an appropriately constructed graph. Thus, the MAP problem can be solved exactly for this class of models in polynomial time. For $L > 2$, the MAP problem

is NP-hard, but a procedure based on a relaxed linear program guarantees a factor 2 approximation of the optimal solution [Boykov *et al.*, 1999b; Kleinberg & Tardos, 1999]. Our associative potentials extend the Potts model in several ways. Importantly, AMNs allow different labels to have different attraction strength: $\phi_{ij}(k,k) = \lambda_{ij}(k)$, where $\lambda_{ij}(k) \geq 1$, and $\phi_{ij}(k,l) = 1, \ \forall k \neq l$. This additional flexibility is important in many domains, as different labels can have very diverse affinities. For example, foreground pixels tend to have locally coherent values while background is much more varied.

In a second important extension, AMNs admit non-pairwise interactions between variables, with potentials over cliques involving m variables $\phi(\mu_{i1}, \ldots, \mu_{im})$. In this case, the clique potentials are constrained to have the same type of structure as the edge potentials: There are K parameters $\phi_c(k, \ldots, k) = \lambda_c(k) \geq 1$ and the rest of the entries are set to 1. In particular, using this additional expressive power, AMNs allow us to encode the pattern of (soft) transitivity present in many domains. For example, consider the problem of predicting whether two proteins interact [Vazquez *et al.*, 2003]; this probability may increase if they *both* interact with another protein. This type of transitivity could be modeled by a ternary clique that has high λ for the assignment with all interactions present.

More formally, we define *associative* functions and potentials as follows.

Definition 7.1.1 *A function $g : \mathcal{Y} \mapsto I\!R$ is* associative for a graph $\mathcal{G}$ over K-ary variables *if it can be written as:*

$$g(\mathbf{y}) = \sum_{v \in \mathcal{V}} \sum_{k=1}^{K} g_v(k) I\!1(y_v = k) + \sum_{c \in \mathcal{C} \setminus \mathcal{V}} \sum_{k=1}^{K} g_c(k) I\!1(\mathbf{y}_c = k, \ldots, k); \qquad g_c(k) \geq 0, \ \ \forall c \in \mathcal{C} \setminus \mathcal{V},$$

where $\mathcal{V}$ are the nodes and $\mathcal{C}$ are the cliques of the graph $\mathcal{G}$. A set of potentials $\phi(\mathbf{y})$ is associative if $\phi(\mathbf{y}) = e^{g(\mathbf{y})}$ and $g(\mathbf{y})$ is associative.

7.2 LP Inference

We can write an integer linear program for the problem of finding the maximum of an associative function $g(\mathbf{y})$, where we have a "marginal" variable $\mu_v(k)$ for each node $v \in \mathcal{V}$ and each label k, which indicates whether node v has value k, and $\mu_c(k)$ for each clique c

(containing more than one variable) and label k, which represents the event that all nodes in the clique c have label k:

$$\begin{aligned} \max \quad & \sum_{v\in\mathcal{V}}\sum_{k=1}^{K}\mu_v(k)g_v(k) + \sum_{c\in\mathcal{C}\setminus\mathcal{V}}\sum_{k=1}^{K}\mu_c(k)g_c(k) \\ \text{s.t.} \quad & \mu_c(k)\in\{0,1\}, \quad \forall c\in\mathcal{C},\ k; \qquad \sum_{k=1}^{K}\mu_v(k)=1, \quad \forall v\in\mathcal{V}; \\ & \mu_c(k)\le\mu_v(k), \quad \forall c\in\mathcal{C}\setminus\mathcal{V},\ v\in c,\ k. \end{aligned} \tag{7.1}$$

Note that we substitute the constraint $\mu_c(k) = \bigwedge_{v\in c}\mu_v(k)$ by linear inequality constraints $\mu_c(k) \le \mu_v(k)$. This works because the coefficient $g_c(k)$ is non-negative and we are maximizing the objective function. Hence at the optimum, $\mu_c(k) = \min_v \mu_v(k)$, which is equivalent to $\mu_c(k) = \bigwedge_{v\in c}\mu_v(k)$, when $\mu_v(k)$ are binary.

It can be shown that in the binary case, the linear relaxation of Eq. (7.1), (where the constraints $\mu_c(k)\in\{0,1\}$ are replaced by $\mu_c(k)\ge 0$), is guaranteed to produce an integer solution when a unique solution exists.

Theorem 7.2.1 *If $K=2$, for any associative function g, the linear relaxation of Eq. (7.1) has an integral optimal solution.*

See Appendix A.2.1 for the proof. This result states that the MAP problem in binary AMNs is tractable, regardless of network topology or clique size. In the non-binary case ($L>2$), these LPs can produce fractional solutions and we use a rounding procedure to get an integral solution.

Theorem 7.2.2 *If $K>2$, for any associative function g, the linear relaxation of Eq. (7.1) has a solution that is larger than the solution of the integer program by at most the number of variables in the largest clique.*

In the appendix, we also show that the approximation ratio of the rounding procedure is the inverse of the size of the largest clique (e.g., $\frac{1}{2}$ for pairwise networks). Although artificial examples with fractional solutions can be easily constructed by using symmetry, it seems that in real data such symmetries are often broken. In fact, in all our experiments with $L>2$ on real data, we never encountered fractional solutions.

7.3 Min-cut inference

We can also use efficient min-cut algorithms to perform exact inference on the learned models for $K = 2$ and approximate inference for $K > 2$. For simplicity, we focus on the pairwise AMN case. We first consider the case of binary AMNs, and later show how to use the local search algorithm developed by Boykov *et al.* [1999a] to perform (approximate) inference in the general multi-class case. For pairwise, binary AMNs, the objective of the integer program in Eq. (7.1) is:

$$\max \sum_{v \in \mathcal{V}} [\mu_v(1) g_v(1) + \mu_v(2) g_v(2)] + \sum_{uv \in \mathcal{E}} [\mu_{uv}(1) g_{uv}(1) + \mu_{uv}(2) g_{uv}(2)]. \tag{7.2}$$

7.3.1 Graph construction

We construct a graph in which the *min-cut* will correspond to the optimal MAP labeling for the above objective. First, we recast the objective as minimization by simply reversing the signs on the value of each θ.

$$\min -\sum_{v \in \mathcal{V}} [\mu_v(1) g_v(1) + \mu_v(2) g_v(2)] - \sum_{uv \in \mathcal{E}} [\mu_{uv}(1) g_{uv}(1) + \mu_{uv}(2) g_{uv}(2)]. \tag{7.3}$$

The graph will consist of a vertex for each node in the AMN, along with the 1 and 2 terminals. In the final $(\mathcal{V}_1, \mathcal{V}_2)$ cut, the $\mathcal{V}_1$ set will correspond to label 1, and the $\mathcal{V}_2$ set will correspond to label 2. We will show how to deal with the node terms (those depending only on a single variable) and the edge terms (those depending on a pair of variables), and then how to combine the two.

Node terms

Consider a node term $-\mu_v(1) g_v(1) - \mu_v(2) g_v(2)$. Such a term corresponds to the node potential contribution to our objective function for node v. For each node term corresponding to node v we add a vertex v to the min-cut graph. We then look at $\Delta_v = g_v(1) - g_v(2)$, and create an edge of weight $|\Delta_v|$ from v to either 1 or 2, depending on the sign of Δ_v. The reason for that is that the final min-cut graph must consist of only positive weights. An

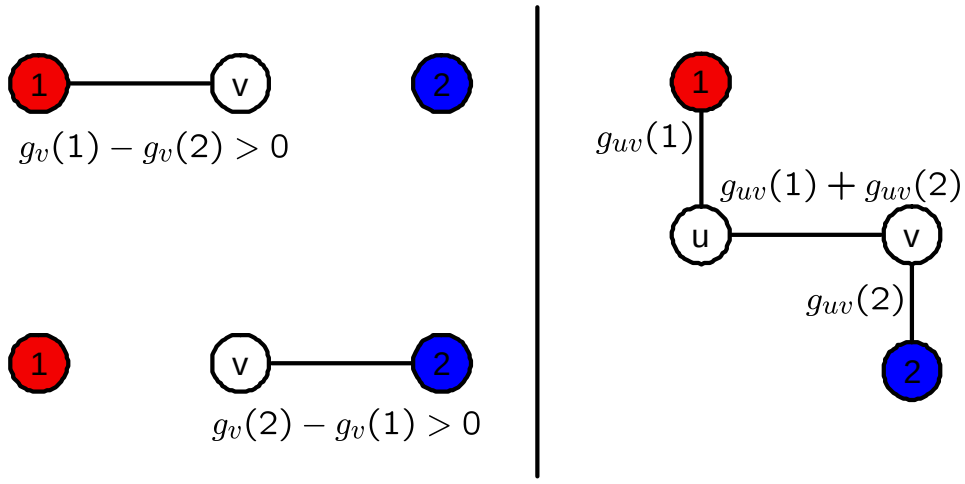


Figure 7.1: Min-cut graph construction of node (left) and edge (right) terms.

example is presented in Fig. 7.3.1.

From Fig. 7.3.1, we see that if the AMN consisted of only node potentials, the graph construction above would add an edge from each node to its more likely label. Thus if we run min-cut, we would simply get a cut with cost 0, since for each introduced vertex we have only one edge of positive weight to either 1 or 2, and we would always choose not to cut any edges.

Edge Terms

Now consider an edge term of the form $-\mu_{uv}(1)g_{uv}(1) - \mu_{uv}(2)g_{uv}(2)$. To construct a min-cut graph for the edge term we will introduce two vertices u and v. We will connect vertex u to 1 with an edge of weight $g_{uv}(1)$, connect v to 2 with an edge of weight $g_{uv}(2)$ and connect u to v with an edge of weight $g_{uv}(1) + g_{uv}(2)$. Fig. 7.3.1 shows an example. Observe what happens when both nodes are on the $\mathcal{V}_2$ side of the cut: the value of the min-cut is $g_{uv}(1)$, which must be less than $g_{uv}(2)$ or the min-cut would have placed them both on the 1 side. When looking at edge terms in isolation, a cut that places each node in different sets will not occur, but when we combine the graphs for node terms and edge terms, such cuts will be possible.

We can take the individual graphs we created for node and edge terms and merge them

by adding edge weights together (and treating missing edges as edges with weight 0). It can be shown that the resulting graph will represent the same objective (in the sense that running min-cut on it will optimize the same objective) as the sum of the objectives of each graph. Since our MAP-inference objective is simply a sum of node and edge terms, merging the node and edge term graphs will result in a graph in which min-cut will correspond to the MAP labeling.

7.3.2 Multi-class case

The graph construction above finds the best MAP labeling for the binary case, but in practice we would often like to handle multiple classes in AMNs. One of the most effective algorithms for minimizing energy functions like ours is the α-expansion algorithm proposed by Boykov *et al.* [1999a]. The algorithm performs a series of "expansion" moves each of which involves optimization over two labels, and it can be shown that it converges to within a factor of 2 of the global minimum.

Expansion Algorithm

Consider a current labeling μ and a particular label $k \in 1, \ldots, K$. Another labeling μ' is called an "α-expansion" move (following Boykov *et al.* [1999a]) from μ if $\mu'_v \neq k$ implies $\mu'_v = \mu_v$ (where μ_v is the label of the node v in the AMN.) In other words, a k-expansion from a current labeling allows each label to either stay the same, or change to k.

The α-expansion algorithm cycles through all labels k in either a fixed or random order, and finds the new labeling whose objective has the lowest value. It terminates when there is no α-expansion move for any label k that has a lower objective than the current labeling (Fig. 7.2).

The key part of the algorithm is computing the best α-expansion labeling for a fixed k and a fixed current labeling μ. The min-cut construction from earlier allows us to do exactly that since an α-expansion move essentially minimizes a MAP-objective over two labels: it either allows a node to retain its current label, or switch to the label α. In this new binary problem we will let label 1 represent a node keeping its current label and label 2 will denote a node taking on the new label k. In order to construct the right coefficients

1. Begin with arbitrary labeling μ
2. Set $success := 0$
3. For each label $k \in \{1, \ldots K\}$
   3.1 Compute $\hat{\mu} = \arg\min -g(\mu')$ among μ' within one α-expansion of μ.
   3.2 If $E(\hat{\mu}) < E(\mu)$, set $\mu := \hat{\mu}$ and success $:= 1$
4. If $success = 1$ goto 2.
5. Return μ

Figure 7.2: α-expansion algorithm

for the new binary objective we need to consider several factors. Below, let $\theta_i'^k$ and $\theta_{ij}'^{k,k}$ denote the node and edge coefficients associated with the new binary objective:

- **Node Potentials** For each node i in the current labeling whose current label is not α, we let $\theta_i'^0 = \theta_i^{y_i}$, and $\theta_i'^1 = \theta_i^{\alpha}$, where y_i denotes the current label of node i, and θ^{y_i} denotes the coefficient in the multiclass AMN MAP objective. Note that we ignore nodes with label α altogether since an α-expansion move cannot change their label.
- **Edge Potentials** For each edge $(i, j) \in E$ whose nodes have labels different from α, we add a new edge potential, with weights $\theta_{ij}'^1 = \theta_{ij}^{\alpha,\alpha}$. If the two nodes of the edge currently have the same label, we set $\theta_{ij}'^0 = \theta_{ij}^{y_i,y_j}$, and if the two nodes currently have different labels we let $\theta_{ij}'^0 = 0$. For each edge $(i, j) \in E$ in which exactly one of the nodes has label α in the current labeling, we add $\theta_{ij}^{\alpha,\alpha}$, to the node potential $\theta_i'^1$ of the node whose label is different from α.

After we have constructed the new binary MAP objective as above, we can apply the min-cut construction from before to get the optimal labeling within one α-expansion from the current one. Veksler [1999] shows that the α-expansion algorithm converges in $O(N)$ iterations where N is the number of nodes. As noted in Boykov *et al.* [1999a] and as we have observed in our experiments, the algorithm terminates only after a few iterations with most of the improvement occurring in the first 2-3 expansion moves.

7.4 Max-margin estimation

The potentials of the AMN are once again log-linear combinations of basis functions. We will need the following assumption to ensure that $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$ is associative:

Assumption 7.4.1 *Basis functions* $\mathbf{f}$ *are component-wise associative for* $\mathcal{G}(\mathbf{x})$ *for any* $(\mathbf{x}, \mathbf{y})$.

Recall that this implies that for cliques larger than one, all basis functions evaluate to 0 for assignments where the values of the nodes are not equal and are non-negative for the assignments where the values of the nodes are equal. To ensure that $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$ is associative, it is useful to separate the basis functions with support only on nodes from those with support on larger cliques.

Definition 7.4.2 *Let* $\dot{\mathbf{f}}$ *be the subset of basis functions* $\mathbf{f}$ *with support only on singleton cliques:*

$$\dot{\mathbf{f}} = \{f \in \mathbf{f} : \forall \mathbf{x} \in \mathcal{X},\ \mathbf{y} \in \mathcal{Y},\ c \in \mathcal{C}(\mathcal{G}(\mathbf{x})),\ |c| > 1,\ f_c(\mathbf{x}_c, \mathbf{y}_c) = 0\}.$$

Let $\ddot{\mathbf{f}} = \mathbf{f} \setminus \dot{\mathbf{f}}$ *be the rest of the basis functions. Let* $\{\dot{\mathbf{w}}, \ddot{\mathbf{w}}\} = \mathbf{w}$ *be the corresponding subsets of parameters.*

It is easy to verify that any non-negative combination of associative functions is associative, and any combination of basis functions with support only on singleton cliques is also associative, so we have:

Lemma 7.4.3 $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$ *is associative for* $\mathcal{G}(\mathbf{x})$ *for any* $(\mathbf{x}, \mathbf{y})$ *whenever Assumption 7.4.1 holds and* $\ddot{\mathbf{w}} \geq 0$.

We must make similar associative assumption on the loss function in order to guarantee that the LP inference can handle it.

Assumption 7.4.4 *The loss function* $\ell(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{y})$ *is associative for* $\mathcal{G}^{(i)}$ *for all* i.

In practice, this restriction is fairly mild, and the Hamming loss, which we use in generalization bounds and experiments, is associative.

Using the above Assumptions 7.4.1 and 7.4.4 and some algebra (see Appendix A.2.3 for derivation), we have the following max-margin QP for AMNs:

$$
\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i,v \in \mathcal{V}^{(i)}} \xi_{i,v} \qquad (7.4)\\
\text{s.t.} \quad & \mathbf{w}^\top \Delta \mathbf{f}_{i,v}(k) - \sum_{c \supset v} m_{i,c,v}(k) \geq \ell_{i,v}(k) - \xi_{i,v}, \quad \forall i, v \in \mathcal{V}^{(i)}, k;\\
& \ddot{\mathbf{w}}^\top \Delta \ddot{\mathbf{f}}_{i,c}(k) + \sum_{v \in c} m_{i,c,v}(k) \geq \ell_{i,c}(k), \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, k;\\
& m_{i,c,v}(k) \geq -\ddot{\mathbf{w}}^\top \ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)})/|c|, \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, v \in c, k;\\
& \ddot{\mathbf{w}} \geq 0;
\end{aligned}
$$

where $\mathbf{f}_{i,c}(k) = \mathbf{f}_{i,c}(k, \ldots, k)$ and $\ell_{i,c}(k) = \ell_{i,c}(k, \ldots, k)$.

While this primal is more complex than the regular M^{3}N factored primal in Eq. (5.4), the basic structure of the first two sets of constraints remains the same: we have local margin requirements and "credit" passed around through messages $m_{i,c,v}(k)$. The extra constraints are due to the associativity constraints on the resulting model.

The dual of Eq. (7.4) (see derivation in Sec. A.2.3) is given by:

$$
\begin{aligned}
\max \quad & \sum_{i,c \in \mathcal{C}^{(i)},\, k} \mu_{i,c}(k) \ell_{i,c}(k) - \frac{C}{2} \left|\left| \sum_{i,v \in \mathcal{V}^{(i)},\, k} \mu_{i,v}(k) \Delta \dot{\mathbf{f}}_{i,v}(k) \right|\right|^2 - \frac{C}{2} \left|\left| \ddot{\nu} + \sum_{i,c \in \mathcal{C}^{(i)},\, k} \mu_{i,c}(k) \Delta \ddot{\mathbf{f}}_{i,c}(k) \right|\right|^2\\
\text{s.t.} \quad & \mu_{i,c}(k) \geq 0, \quad \forall i,\ \forall c \in \mathcal{C}^{(i)},\ k; \qquad \sum_{k=1}^{K} \mu_{i,v}(k) = 1, \quad \forall i,\ \forall v \in \mathcal{V}^{(i)};\\
& \mu_{i,c}(k) \leq \mu_{i,v}(k), \quad \forall i,\ \forall c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)},\ v \in c,\ k;\\
& \ddot{\nu} \geq 0.
\end{aligned}
$$

In the dual, there are marginals μ for each node and clique, for each value k, similar to Eq. (5.12). However, the constraints are different, and not surprisingly, are essentially the constraints from the inference LP relaxation in Eq. (7.1).

The dual and primal solutions are related by

$$\dot{\mathbf{w}} = \sum_{i,v\in\mathcal{V}^{(i)},\,k} \mu_{i,v}(k)\Delta\dot{\mathbf{f}}_{i,v}(k); \qquad \ddot{\mathbf{w}} = \ddot{\nu} + \sum_{i,c\in\mathcal{C}^{(i)},\,k} \mu_{i,c}(k)\Delta\ddot{\mathbf{f}}_{i,c}(k).$$

The $\ddot{\nu}$ variables simply ensure that $\ddot{\mathbf{w}}$ are positive (if any component $\sum_{i,c\in\mathcal{C}^{(i)},\,k} \mu_{i,c}(k)\Delta\ddot{\mathbf{f}}_{i,c}(k)$ is negative, maximizing the objective will force the corresponding component of $\ddot{\nu}$ to cancel it out). Note that the objective can be written in terms of dot products of node basis functions $\Delta\dot{\mathbf{f}}_{i,v}(k)^\top\Delta\dot{\mathbf{f}}_{j,\bar{v}}(\bar{k})$, so they can be kernelized. Unfortunately, the edge basis functions cannot be kernelized because of the non-negativity constraint.

For $K = 2$, the LP inference is exact, so that Eq. (7.4) learns *exact* max-margin weights for Markov networks of *arbitrary* topology. For $K > 2$, the linear relaxation leads to a strengthening of the constraints on $\mathbf{w}$ by potentially adding constraints corresponding to fractional assignments as in the case of untriangualated networks. Thus, the optimal choice $\mathbf{w}, \xi$ for the original QP may no longer be feasible, leading to a different choice of weights. However, as our experiments show, these weights tend to do well in practice.

7.5 Experiments

We applied associative Markov networks to the task of terrain classification. Terrain classification is very useful for autonomous mobile robots in real-world environments for path planning, target detection, and as a pre-processing step for other perceptual tasks. The Stanford Segbot Project[1] has provided us with a laser range maps of the Stanford campus collected by a moving robot equipped with SICK2 laser sensors Fig. 7.5. The data consists of around 35 million points, represented as 3D coordinates in an absolute frame of reference Fig. 7.5. Thus, the only available information is the location of points. Each reading was a point in 3D space, represented by its (x, y, z) coordinates in an absolute frame of reference. Thus, the only available information is the location of points, which was fairly noisy because of localization errors.

Our task is to classify the laser range points into four classes: ground, building, tree,

[1]Many thanks to Michael Montemerlo and Sebastian Thrun for sharing the data.

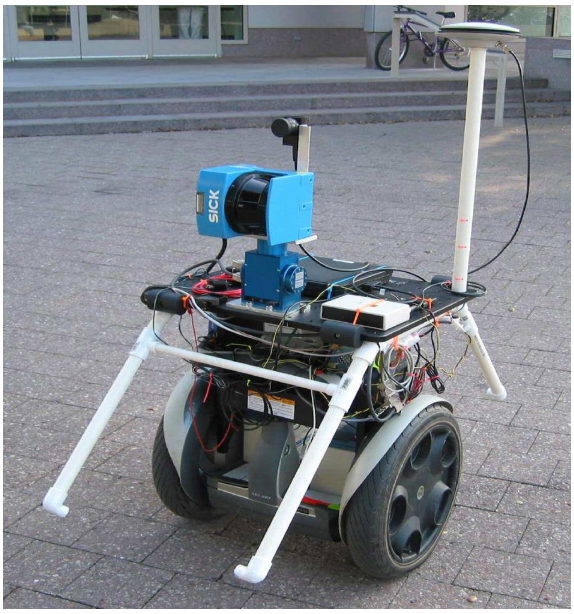


Figure 7.3: Segbot: roving robot equipped with SICK2 laser sensors.

and shrubbery. Since classifying ground points is trivial given their absolute z-coordinate (height), we classify them deterministically by thresholding the z coordinate at a value close to 0. After we do that, we are left with approximately 20 million non-ground points. Each point is represented simply as a location in an absolute 3D coordinate system. The features we use require pre-processing to infer properties of the local neighborhood of a point, such as how planar the neighborhood is, or how much of the neighbors are close to the ground. The features we use are invariant to rotation in the x-y plane, as well as the density of the range scan, since scans tend to be sparser in regions farther from the robot.

Our first type of feature is based on the principal plane around it. For each point we sample 100 points in a cube of radius 0.5 meters. We run PCA on these points to get the plane of maximum variance (spanned by the first two principal components). We then partition the cube into $3 \times 3 \times 3$ bins around the point, oriented with respect to the principal plane, and compute the percentage of points lying in the various sub-cubes. We use a number of features derived from the cube such as the percentage of points in the central column, the outside corners, the central plane, etc. These features capture the local distribution well and are especially useful in finding planes. Our second type of feature is based on a column

Figure 7.4: 3D laser scan range map of the Stanford Quad.

around each point. We take a cylinder of radius 0.25 meters, which extends vertically to include all the points in a "column". We then compute what percentage of the points lie in various segments of this vertical column (e.g., between 2m and 2.5m). Finally, we also use an indicator feature of whether or not a point lies within $2m$ of the ground. This feature is especially useful in classifying shrubbery.

For training we select roughly 30 thousand points that represent the classes well: a segment of a wall, a tree, some bushes. We considered three different models: SVM, Voted-SVM and AMNs. All methods use the same set of features, augmented with a quadratic kernel.

The first model is a multi-class SVM with a quadratic kernel over the above features. This model (Fig. 7.5, right panel and Fig. 7.7, top panel) achieves reasonable performance

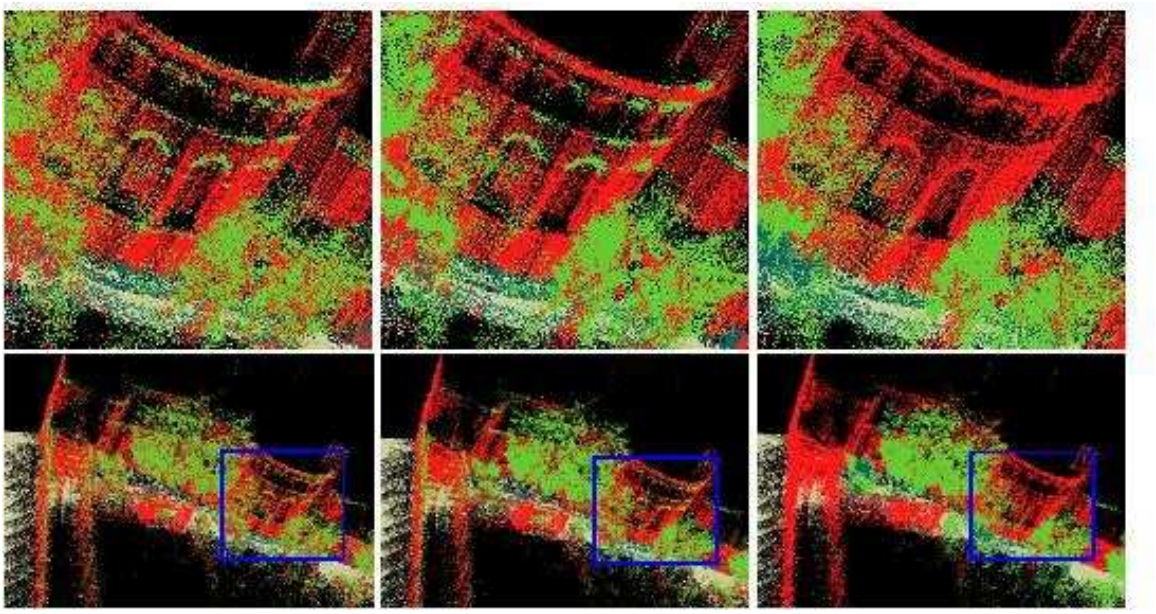


Figure 7.5: Terrain classification results showing Stanford Memorial Church obtained with SVM, Voted-SVM and AMN models. (Color legend: buildings/red, trees/green, shrubs/blue, ground/gray).

in many places, but fails to enforce local consistency of the classification predictions. For example arches on buildings and other less planar regions are consistently confused for trees, even though they are surrounded entirely by buildings.

We improved upon the SVM by smoothing its predictions using voting. For each point we took its local neighborhood (we varied the radius to get the best possible results) and assigned the point the label of the majority of its 100 neighbors. The Voted-SVM model (Fig. 7.5, middle panel and Fig. 7.7, middle panel) performs slightly better than SVM: for example, it smooths out trees and some parts of the buildings. Yet it still fails in areas like arches of buildings where the SVM classifier has a locally consistent wrong prediction.

The final model is a pairwise AMN over laser scan points, with associative potentials to ensure smoothness. Each point is connected to 6 of its neighbors: 3 of them are sampled randomly from the local neighborhood in a sphere of radius 0.5m, and the other 3 are sampled at random from the vertical cylinder column of radius 0.25m. It is important to ensure vertical consistency since the SVM classifier is wrong in areas that are higher off the

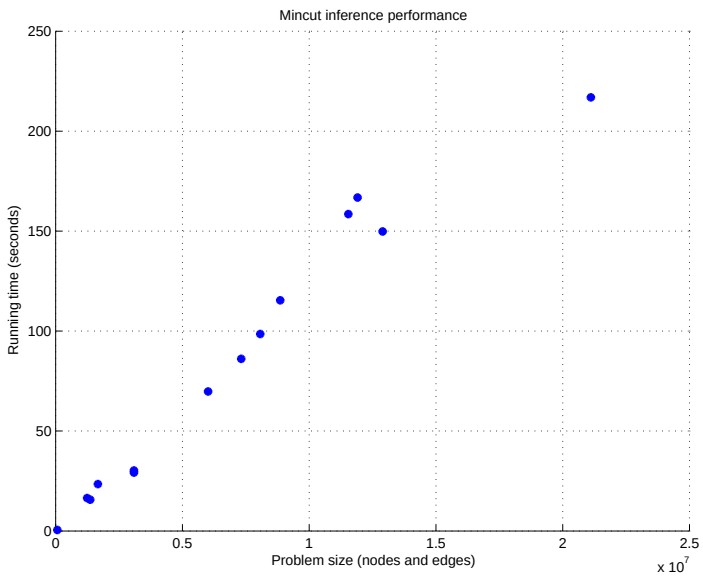


Figure 7.6: The running time (in seconds) of the min-cut-based inference algorithm for different problem sizes. The problem size is the sum of the number of nodes and the number of edges. Note the near linear performance of the algorithm and its efficiency even for large models.

ground (due to the decrease in point density) or because objects tend to look different as we vary their z-coordinate (for example, tree trunks and tree crowns look different). While we experimented with a variety of edge features including various distances between points, we found that even using only a constant feature performs well.

We trained the AMN model using CPLEX to solve the quadratic program; the training took about an hour on a Pentium 3 desktop. The inference over each segment was performed using min-cut with α-expansion moves as described above. We used a publicly available implementation of the min-cut algorithm, which uses bidirectional search trees for augmenting paths (see Boykov and Kolmogorov [2004]). The implementation is largely dominated by I/O time, with the actual min-cut taking less than two minutes even for the largest segment. The performance is summarized in Fig. 7.6, and as we can see, it is roughly linear in the size of the problem (number of nodes and number of edges).

We can see that the predictions of the AMN (Fig. 7.5, left panel and Fig. 7.7, bottom panel) are much smoother: for example building arches and tree trunks are predicted

correctly. We also hand-labeled around 180 thousand points of the test set (Fig. 7.8) and computed accuracies of the predictions shown in Fig. 7.9 (excluding ground, which was classified by pre-processing). The differences are dramatic: SVM: 68%, Voted-SVM: 73% and AMN: 93%. See more results, including a fly-through movie of the data, at `http://ai.stanford.edu/~btaskar/3Dmap/`.

7.6 Related work

Several authors have considered extensions to the Potts model. Kleinberg and Tardos [1999] extend the multi-class Potts model to have more general edge potentials, under the constraints that negative log of the edge potentials form a metric on the set of labels. They also provide a solution based on a relaxed LP that has certain approximation guarantees.

More recently, Kolmogorov and Zabih [2002] showed how to optimize energy functions containing binary and ternary interactions using graph cuts, as long as the parameters satisfy a certain regularity condition. Our definition of associative potentials below also satisfies the Kolmogorov and Zabih regularity condition for $K = 2$. However, the structure of our potentials is simpler to describe and extend for the multi-class case. In fact, we can extend our max-margin framework to estimate their more general potentials by expressing inference as a linear program.

Our terrain classification approach is most closely related to work in vision applying conditional random fields (CRFs) to 2D images. Kumar and Hebert [2003] train CRFs using a pseudo-likelihood approximation to the distribution $P(\mathbf{Y} \mid \mathbf{X})$ since estimating the true conditional distribution is intractable. Unlike their work, our learning formulation provides an exact and tractable optimization algorithm, as well as formal guarantees for binary classification problems. Moreover, unlike their work, our approach can also handle multi-class problems in a straightforward manner.

7.7 Conclusion

In this chapter, we provide an algorithm for max-margin training of associative Markov networks, a subclass of Markov networks that allows only positive interactions between

related variables. Our approach relies on a linear programming relaxation of the MAP problem, which is the key component in the quadratic program associated with the max-margin formulation. We thus provide a polynomial time algorithm which approximately solves the maximum margin estimation problem for any associative Markov network. Importantly, our method is guaranteed to find the optimal (margin-maximizing) solution for all binary-valued AMNs, regardless of the clique size or the connectivity. To our knowledge, this algorithm is the first to provide an effective learning procedure for Markov networks of such general structure.

Our results in the binary case rely on the fact that the LP relaxation of the MAP problem provides exact solutions. In the non-binary case, we are not guaranteed exact solutions, but we can prove constant-factor approximation bounds on the MAP solution returned by the relaxed LP. It would be interesting to see whether these bounds provide us with guarantees on the quality (e.g., the margin) of our learned model.

We present large-scale experiments with terrain segmentation and classification from 3D range data involving AMNs with tens of millions of nodes and edges. The class of associative Markov networks appears to cover a large number of interesting applications. We have explored only a computer vision application in this chapter, and consider another one (hypertext classification) in the next. It would be very interesting to consider other applications, such as extracting protein complexes from protein-protein interaction data, or predicting links in relational data. The min-cut based inference is able to handle very large networks, and it is an interesting challenge to apply the algorithm to even larger models and develop efficient distributed implementations.

However, despite the prevalence of fully associative Markov networks, it is clear that many applications call for repulsive potentials. While clearly we cannot introduce fully general potentials into AMNs without running against the NP-hardness of the general problem, it would be interesting to see whether we can extend the class of networks we can learn effectively.

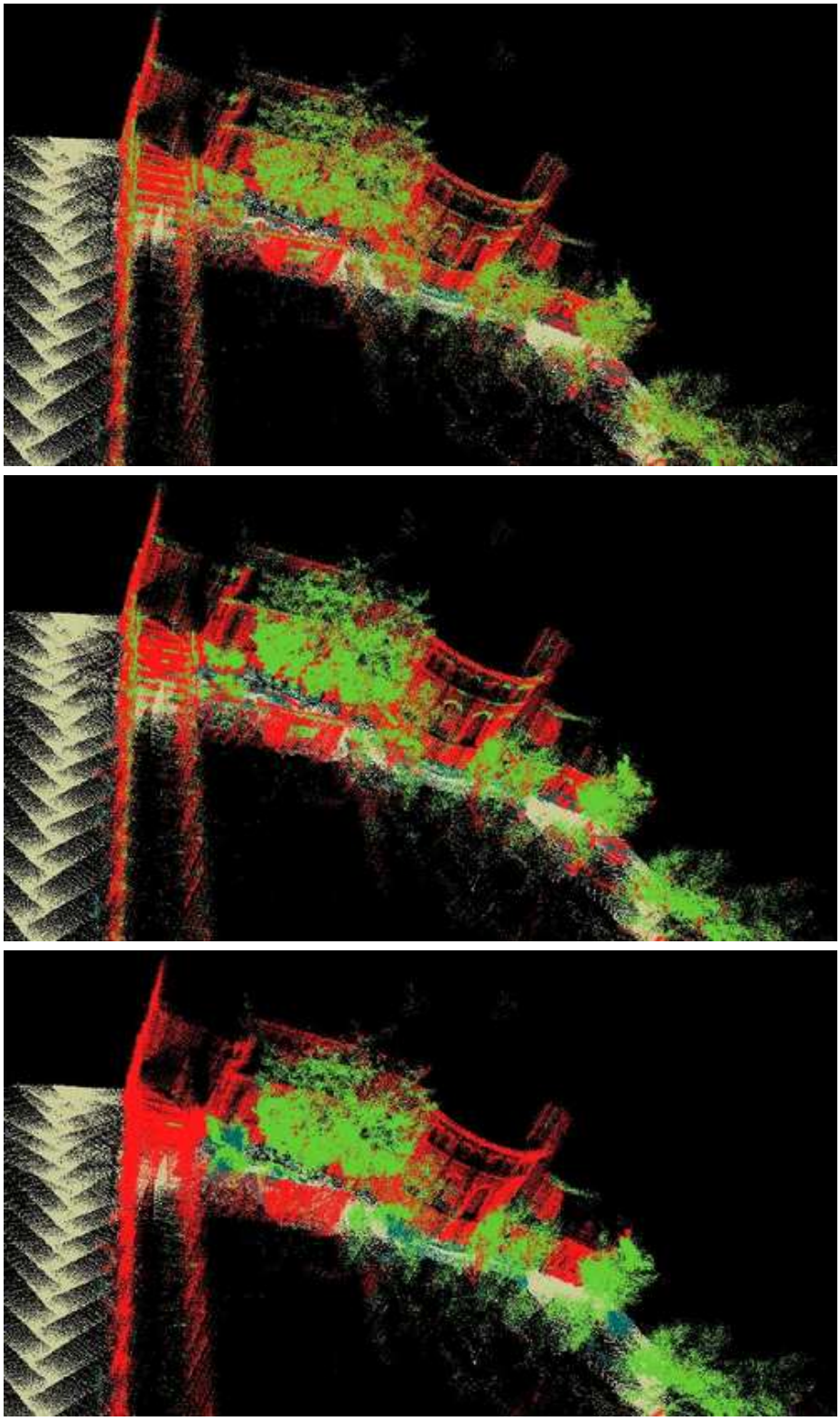


Figure 7.7: Results from the SVM, Voted-SVM and AMN models.

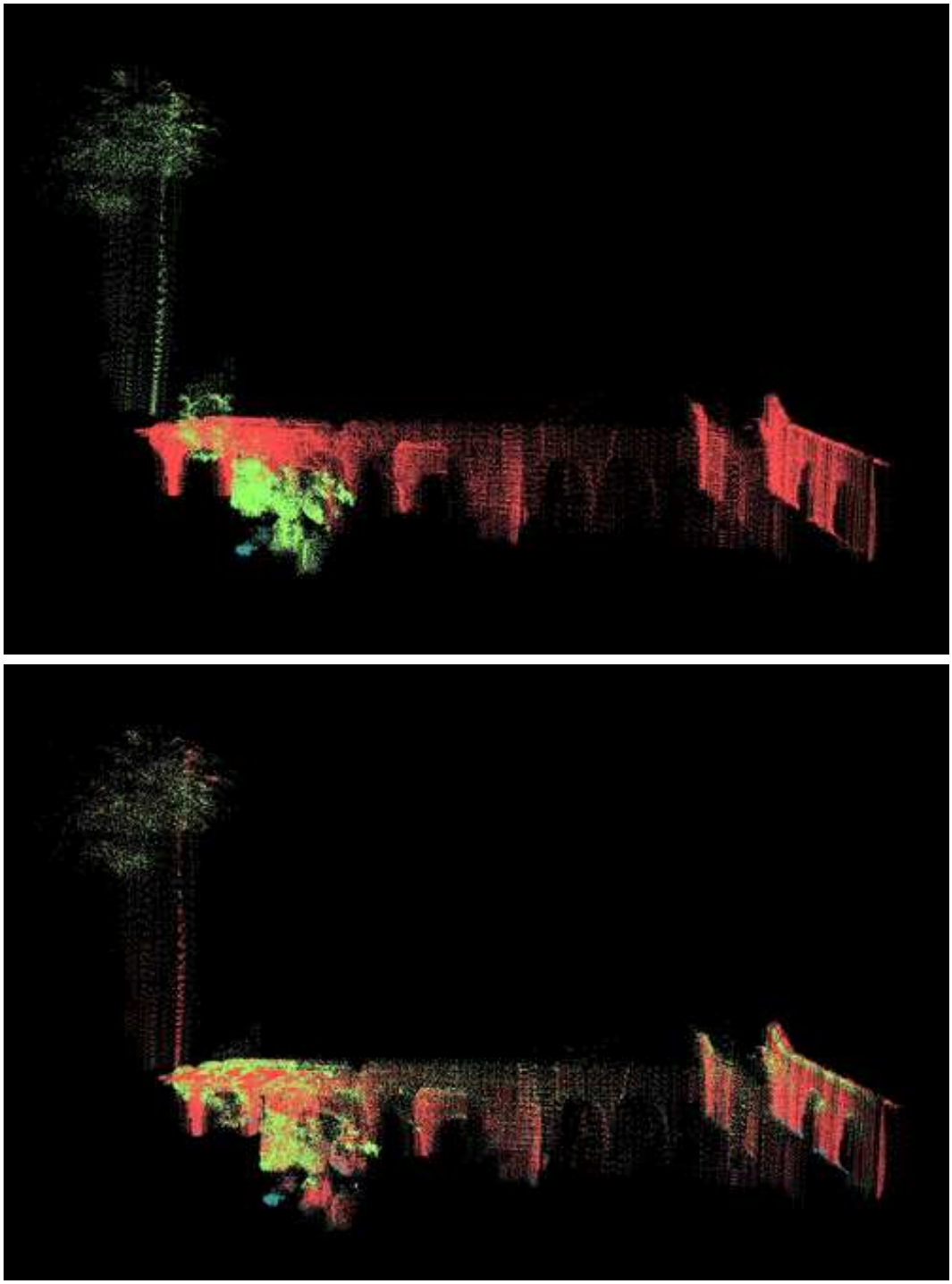


Figure 7.8: Labeled part of the test set: ground truth (top) and SVM predictions (bottom).

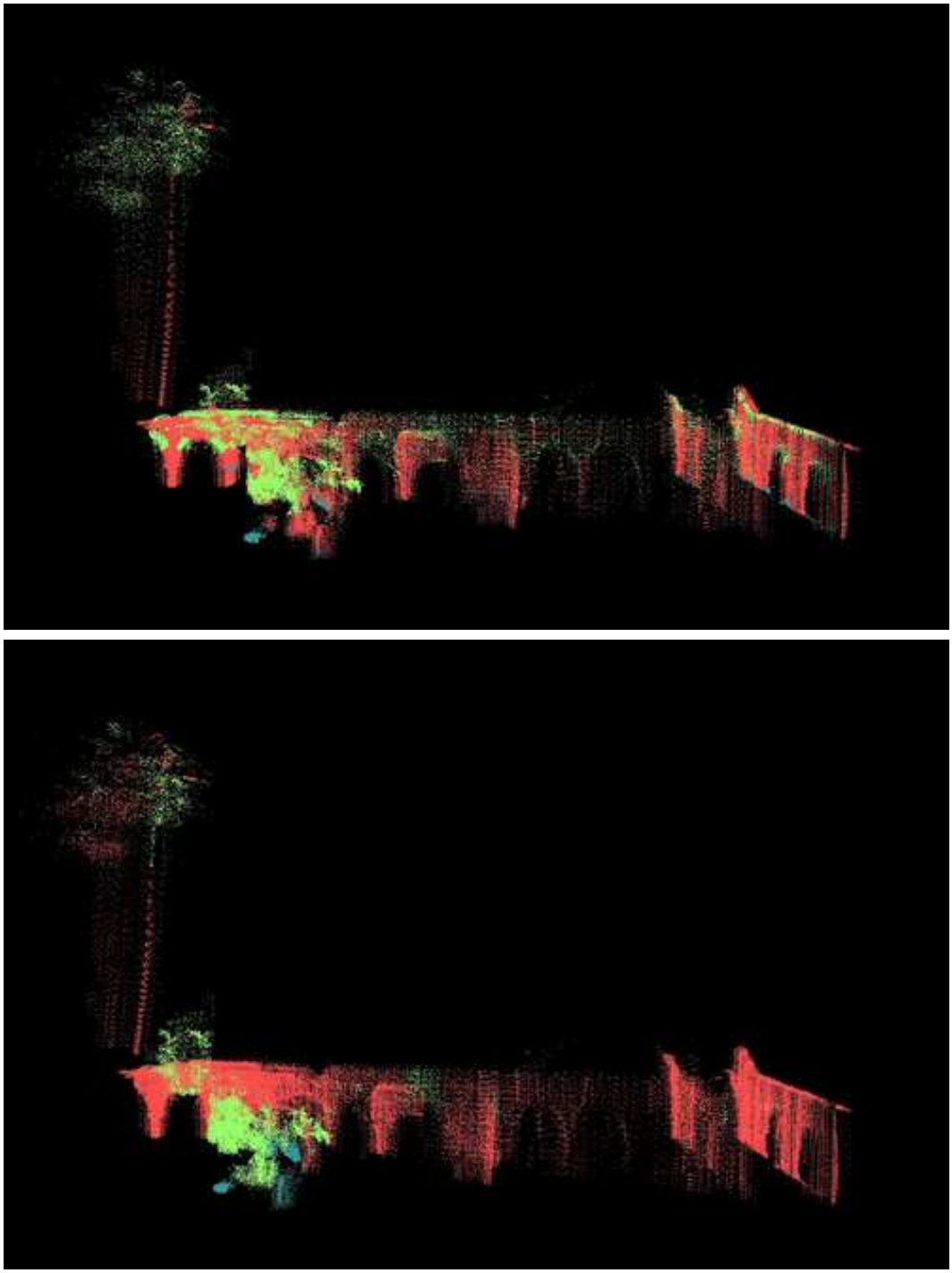


Figure 7.9: Predictions of the Voted-SVM (top) and AMN (bottom) models.

Chapter 8

Relational Markov networks

In the previous chapters, we have seen how sequential and spatial correlation between labels can be exploited for tremendous accuracy gains. In many other supervised learning tasks, the entities to be labeled are related with each other in very complex ways, not just sequentially or spatially. For example, in hypertext classification, the labels of linked pages are highly correlated. A standard approach is to classify each entity independently, ignoring the correlations between them. In this chapter, we present a framework that builds on Markov networks and provides a flexible language for modeling rich interaction patterns in structured data. We provide experimental results on a webpage classification task, showing that accuracy can be significantly improved by modeling relational dependencies.

Many real-world data sets are innately relational: hyperlinked webpages, cross-citations in patents and scientific papers, social networks, medical records, and more. Such data consist of entities of different types, where each entity type is characterized by a different set of attributes. Entities are related to each other via different types of links, and the link structure is an important source of information.

Consider a collection of hypertext documents that we want to classify using some set of labels. Most naively, we can use a bag of words model, classifying each webpage solely using the words that appear on the page. However, hypertext has a very rich structure that this approach loses entirely. One document has hyperlinks to others, typically indicating that their topics are related. Each document also has internal structure, such as a partition into sections; hyperlinks that emanate from the same section of the document are even

more likely to point to similar documents. When classifying a collection of documents, these are important cues, that can potentially help us achieve better classification accuracy. Therefore, rather than classifying each document separately, we want to provide a form of *collective classification*, where we simultaneously decide on the class labels of all of the entities together, and thereby can explicitly take advantage of the correlations between the labels of related entities.

We propose the use of a joint probabilistic model for an entire collection of related entities. We introduce the framework of *relational Markov networks (RMNs)*, which compactly defines a Markov network over a relational data set. The graphical structure of an RMN is based on the relational structure of the domain, and can easily model complex patterns over related entities. For example, we can represent a pattern where two linked documents are likely to have the same topic. We can also capture patterns that involve groups of links: for example, consecutive links in a document tend to refer to documents with the same label. As we show, the use of an undirected graphical model avoids the difficulties of defining a coherent generative model for graph structures in directed models. It thereby allows us tremendous flexibility in representing complex patterns.

8.1 Relational classification

Consider hypertext as a simple example of a relational domain. A relational domain is defined by a schema, which describes entities, their attributes and relations between them. In our domain, there are two entity types: Doc and Link. If a webpage is represented as a bag of words, Doc would have a set of boolean attributes Doc.*HasWord*$_k$ indicating whether the word k occurs on the page. It would also have the label attribute Doc.*Label*, indicating the topic of the page, which takes on a set of categorical values. The Link entity type has two attributes: Link.*From* and Link.*To*, both of which refer to Doc entities.

In general, a *schema* specifies of a set of entity types $\mathcal{E} = \{E_1, \ldots, E_n\}$. Each type E is associated with three sets of attributes: content attributes $E.\mathbf{X}$ (for example, Doc.*HasWord*$_k$), label attributes $E.\mathbf{Y}$ (for example, Doc.*Label*), and reference attributes $E.\mathbf{R}$ (for example, Link.*To*). For simplicity, we restrict label and content attributes to take on categorical values. Reference attributes include a special unique key attribute $E.K$ that identifies each

entity. Other reference attributes $E.R$ refer to entities of a single type $E' = Range(E.R)$ and take values in $Domain(E'.K)$.

An *instantiation* $\mathcal{I}$ of a schema $\mathcal{E}$ specifies the set of entities $\mathcal{I}(E)$ of each entity type $E \in \mathcal{E}$ and the values of all attributes for all of the entities. For example, an instantiation of the hypertext schema is a collection of webpages, specifying their labels, words they contain and links between them. We will use $\mathcal{I}.\mathbf{X}$, $\mathcal{I}.\mathbf{Y}$ and $\mathcal{I}.\mathbf{R}$ to denote the content, label and reference attributes in the instantiation $\mathcal{I}$; $\mathcal{I}.\mathbf{x}$, $\mathcal{I}.\mathbf{y}$ and $\mathcal{I}.\mathbf{r}$ to denote the values of those attributes. The component $\mathcal{I}.\mathbf{r}$, which we call an *instantiation skeleton* or *instantiation graph*, specifies the set of entities (nodes) and their reference attributes (edges). A hypertext instantiation graph specifies a set of webpages and links between them, but not their words or labels. Taskar *et al.* [2001] suggest the use of *probabilistic relational models (PRMs)* for the collective classification task. PRMs [Koller & Pfeffer, 1998; Friedman *et al.*, 1999; Getoor *et al.*, 2002] are a relational extension of Bayesian networks [Pearl, 1988]. A PRM specifies a probability distribution over instantiations consistent with a given instantiation graph by specifying a Bayesian-network-like template-level probabilistic model for each entity type. Given a particular instantiation graph, the PRM induces a large Bayesian network over that instantiation that specifies a joint probability distribution over all attributes of all of the entities. This network reflects the interactions between related instances by allowing us to represent correlations between their attributes.

In our hypertext example, a PRM might use a naive Bayes model for words, with a directed edge between $\mathsf{Doc}.\mathit{Label}$ and each attribute $\mathsf{Doc}.\mathit{HadWord}_k$; each of these attributes would have a *conditional probability distribution* $P(\mathsf{Doc}.\mathit{HasWord}_k \mid \mathsf{Doc}.\mathit{Label})$ associated with it, indicating the probability that word k appears in the document given each of the possible topic labels. More importantly, a PRM can represent the inter-dependencies between topics of linked documents by introducing an edge from $\mathsf{Doc}.\mathit{Label}$ to $\mathsf{Doc}.\mathit{Label}$ of two documents if there is a link between them. Given a particular instantiation graph containing some set of documents and links, the PRM specifies a Bayesian network over all of the documents in the collection. We would have a probabilistic dependency from each document's label to the words on the document, and a dependency from each document's label to the labels of all of the documents to which it points. Taskar *et al.* show that this approach works well for classifying scientific documents, using both the words in the title

and abstract and the citation-link structure.

However the application of this idea to other domains, such as webpages, is problematic since there are many cycles in the link graph, leading to cycles in the induced "Bayesian network", which is therefore not a coherent probabilistic model. Getoor *et al.* [2001] suggest an approach where we do not include direct dependencies between the labels of linked webpages, but rather treat links themselves as random variables. Each two pages have a "potential link", which may or may not exist in the data. The model defines the probability of the link existence as a function of the labels of the two endpoints. In this link existence model, labels have no incoming edges from other labels, and the cyclicity problem disappears. This model, however, has other fundamental limitations. In particular, the resulting Bayesian network has a random variable for each potential link — N^2 variables for collections containing N pages. This quadratic blowup occurs even when the actual link graph is very sparse. When N is large (e.g., the set of all webpages), a quadratic growth is intractable. Even more problematic are the inherent limitations on the expressive power imposed by the constraint that the directed graph must represent a coherent generative model over graph structures. The link existence model assumes that the presence of different edges is a conditionally independent event. Representing more complex patterns involving correlations between multiple edges is very difficult. For example, if two pages point to the same page, it is more likely that they point to each other as well. Such interactions between many overlapping triples of links do not fit well into the generative framework.

Furthermore, directed models such as Bayesian networks and PRMs are usually trained to optimize the joint probability of the labels and other attributes, while the goal of classification is a discriminative model of labels given the other attributes. The advantage of training a model only to discriminate between labels is that it does not have to trade off between classification accuracy and modeling the joint distribution over non-label attributes. In many cases, discriminatively trained models are more robust to violations of independence assumptions and achieve higher classification accuracy than their generative counterparts.

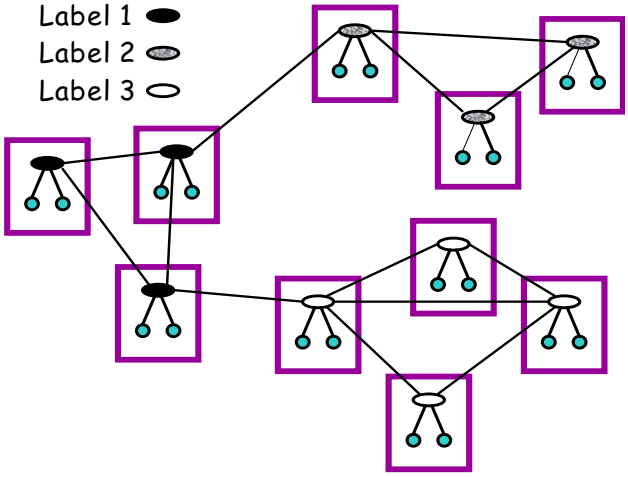


Figure 8.1: An unrolled Markov net over linked documents. The links follow a common pattern: documents with the same label tend to link to each other more often.

8.2 Relational Markov networks

We now extend the framework of Markov networks to the relational setting. A *relational Markov network (RMN)* specifies a conditional distribution over all of the labels of all of the entities in an instantiation given the relational structure and the content attributes. (We provide the definitions directly for the conditional case, as the unconditional case is a special case where the set of content attributes is empty.) Roughly speaking, it specifies the cliques and potentials between attributes of related entities at a template level, so a single model provides a coherent distribution for any collection of instances from the schema.

For example, suppose that pages with the same label tend to link to each other, as in Fig. 8.1. We can capture this correlation between labels by introducing, for each link, a clique between the labels of the source and the target page. The potential on the clique will have higher values for assignments that give a common label to the linked pages.

To specify what cliques should be constructed in an instantiation, we will define a notion of a *relational clique template*. A relational clique template specifies tuples of variables in the instantiation by using a relational query language. For our link example, we can write the template as a kind of SQL query:

```
SELECT doc1.Category, doc2.Category
```

FROM Doc doc1, Doc doc2, Link link
WHERE link.From = doc1.Key and link.To = doc2.Key

Note the three clauses that define a query: the FROM clause specifies the cross product of entities to be filtered by the WHERE clause and the SELECT clause picks out the attributes of interest. Our definition of clique templates contains the corresponding three parts.

A *relational clique template* $C = (\mathbf{F}, \mathbf{W}, \mathbf{S})$ consists of three components:

- $\mathbf{F} = \{F_i\}$ — a set of entity variables, where an entity variable F_i is of type $E(F_i)$.
- $\mathbf{W}(\mathbf{F}.\mathbf{R})$ — a boolean formula using conditions of the form $F_i.R_j = F_k.R_l$.
- $\mathbf{F}.\mathbf{S} \subseteq \mathbf{F}.\mathbf{X} \cup \mathbf{F}.\mathbf{Y}$ — a selected subset of content and label attributes in $\mathbf{F}$. ∎

For the clique template corresponding to the SQL query above, $\mathbf{F}$ consists of *doc1*, *doc2* and *link* of types Doc, Doc and Link, respectively. $\mathbf{W}(\mathbf{F}.\mathbf{R})$ is $link.From = doc1.Key \wedge link.To = doc2.Key$ and $\mathbf{F}.\mathbf{S}$ is $doc1.Category$ and $doc2.Category$.

A clique template specifies a set of cliques in an instantiation $\mathcal{I}$:

$$C(\mathcal{I}) \equiv \{c = \mathbf{f}.\mathbf{S} : \mathbf{f} \in \mathcal{I}(\mathbf{F}) \wedge \mathbf{W}(\mathbf{f}.\mathbf{r})\},$$

where $\mathbf{f}$ is a tuple of entities $\{f_i\}$ in which each f_i is of type $E(F_i)$; $\mathcal{I}(\mathbf{F}) = \mathcal{I}(E(F_1)) \times \ldots \times \mathcal{I}(E(F_n))$ denotes the cross-product of entities in the instantiation; the clause $\mathbf{W}(\mathbf{f}.\mathbf{r})$ ensures that the entities are related to each other in specified ways; and finally, $\mathbf{f}.\mathbf{S}$ selects the appropriate attributes of the entities. Note that the clique template does not specify the nature of the interaction between the attributes; that is determined by the clique potentials, which will be associated with the template.

This definition of a clique template is very flexible, as the WHERE clause of a template can be an arbitrary predicate. It allows modeling complex relational patterns on the instantiation graphs. To continue our webpage example, consider another common pattern in hypertext: links in a webpage tend to point to pages of the same category. This pattern can be expressed by the following template:

SELECT doc1.Category, doc2.Category

```
FROM Doc doc1, Doc doc2, Link link1, Link link2
WHERE link1.From = link2.From and link1.To = doc1.Key
        and link2.To = doc2.Key and not doc1.Key = doc2.Key
```

Depending on the expressive power of our template definition language, we may be able to construct very complex templates that select entire subgraph structures of an instantiation. We can easily represent patterns involving three (or more) interconnected documents without worrying about the acyclicity constraint imposed by directed models. Since the clique templates do not explicitly depend on the identities of entities, the same template can select subgraphs whose structure is fairly different. The RMN allows us to associate the same clique potential parameters with all of the subgraphs satisfying the template, thereby allowing generalization over a wide range of different structures.

A *Relational Markov network (RMN)* $\mathcal{M} = (\mathbf{C}, \Phi)$ specifies a set of clique templates $\mathbf{C}$ and corresponding potentials $\Phi = \{\phi_C\}_{C \in \mathbf{C}}$ to define a conditional distribution:

$$\begin{aligned} & P(\mathcal{I}.\mathbf{y} \mid \mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) \\ & \quad = \frac{1}{Z(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r})} \prod_{C \in \mathbf{C}} \prod_{c \in C(\mathcal{I})} \phi_C(\mathcal{I}.\mathbf{x}_c, \mathcal{I}.\mathbf{y}_c) \end{aligned}$$

where $Z(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r})$ is the normalizing partition function:

$$Z(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) = \sum_{\mathcal{I}.\mathbf{y}'} \prod_{C \in \mathbf{C}} \prod_{c \in C(\mathcal{I})} \phi_C(\mathcal{I}.\mathbf{x}_c, \mathcal{I}.\mathbf{y}'_c)$$

.

Using the log-linear representation of potentials, $\phi_C(\mathbf{V}_C) = \exp\{\mathbf{w}_C^\top \mathbf{f}_C(\mathbf{V}_C)\}$, we can write

$$\log P(\mathcal{I}.\mathbf{y} \mid \mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r}) \quad = \quad \mathbf{w}^\top \mathbf{f}(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{y}, \mathcal{I}.\mathbf{r}) - \log Z(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{r})$$

where

$$\mathbf{f}_C(\mathcal{I}.\mathbf{x}, \mathcal{I}.\mathbf{y}, \mathcal{I}.\mathbf{r}) = \sum_{c \in C(\mathcal{I})} \mathbf{f}_C(\mathcal{I}.\mathbf{x}_c, \mathcal{I}.\mathbf{y}_c)$$

is the sum over all appearances of the template $C(\mathcal{I})$ in the instantiation, and $\mathbf{f}$ is the vector

of all $\mathbf{f}_C$.

Given a particular instantiation $\mathcal{I}$ of the schema, the RMN $\mathcal{M}$ produces an *unrolled* Markov network over the attributes of entities in $\mathcal{I}$. The cliques in the unrolled network are determined by the clique templates C. We have one clique for each $c \in C(\mathcal{I})$, and all of these cliques are associated with the same clique potential ϕ_C. In our webpage example, an RMN with the link basis function described above would define a Markov net in which, for every link between two pages, there is an edge between the labels of these pages. Fig. 8.1 illustrates a simple instance of this unrolled Markov network.

8.3 Approximate inference and learning

Applying both maximum likelihood and maximum margin learning in the relational setting is requires inference in very large and complicated networks, where exact inference is typically intractable. We therefore resort to approximate methods.

Maximum likelihood estimation

For maximum likelihood learning, we need to compute basis function expectations, not just the most likely assignment. There is a wide variety of approximation schemes for this problem, including MCMC and variational methods. We chose to use *belief propagation* for its simplicity and relative efficiency and accuracy. Belief Propagation (BP) is a local message passing algorithm introduced by Pearl [1988]. It is guaranteed to converge to the correct marginal probabilities for each node only for singly connected Markov networks. However, recent analysis [Yedidia *et al.*, 2000] provides some theoretical justification. Empirical results [Murphy *et al.*, 1999] show that it often converges in general networks, and when it does, the marginals are a good approximation to the correct posteriors. As our results in Sec. 8.4 show, this approach works well in our domain. We refer the reader to Yedidia *et al.* for a detailed description of the BP algorithm.

We provide a brief outline of one variant of BP, referring to [Murphy *et al.*, 1999] for more details. For simplicity, we assume a pairwise network where all potentials are associated only with nodes and edges given by:

$$P(Y_1, \ldots, Y_n) = \frac{1}{Z} \prod_{ij} \psi_{ij}(Y_i, Y_j) \prod_i \psi_i(Y_i)$$

where ij ranges over the edges of the network and $\psi_{ij}(Y_i, Y_j) = \phi(\mathbf{x}_{ij}, Y_i, Y_j)$, $\psi_i(Y_i) = \phi(\mathbf{x}_i, Y_i)$.

The belief propagation algorithm is very simple. At each iteration, each node Y_i sends the following messages to all its neighbors $N(i)$:

$$m_{ij}(Y_j) \leftarrow \alpha \sum_{y_i} \psi_{ij}(y_i, Y_j)\psi_i(y_i) \prod_{k \in N(i)-j} m_{ki}(Y_i)$$

where α is a (different) normalizing constant. This process is repeated until the messages converge. At any point in the algorithm, the marginal distribution of any node Y_i is approximated by

$$b_i(Y_i) = \alpha \psi_i(Y_i) \prod_{k \in N(i)} m_{ki}(Y_i)$$

and the marginal distribution of a pair of nodes connected by an edge is approximated by

$$b_{ij}(Y_i, Y_j) = \alpha \psi_{ij}(Y_i, Y_j)\psi_i(Y_i)\psi_j(Y_j) \prod_{k \in N(i)-j} m_{ki}(Y_i) \prod_{l \in N(j)-i} m_{lj}(Y_j)$$

These approximate marginals are precisely what we need for the computation of the basis function expectations and performing classification. Computing the expected basis function expectations involves summing their expected values for each clique using the approximate marginals $b_i(Y_i)$ and $b_{ij}(Y_i, Y_j)$. Similarly, we use $\max_{y_i} b_i(Y_i)$ at prediction time. Note that we can also $max - product$ variant of loopy BP, with

$$m_{ij}(Y_j) \leftarrow \alpha \max_{y_i} \psi_{ij}(y_i, Y_j)\psi_i(y_i) \prod_{k \in N(i)-j} m_{ki}(Y_i)$$

to compute approximate posterior “max”-marginals and use those for prediction. In our experiments, this results in less accurate classification, so we use posterior marginal prediction.

Maximum margin estimation

For maximum margin estimation, we used approximate LP inference inside the max-margin QP, using commercial Ilog CPLEX software to solve it. For networks with general potentials, we used the untriangulated LP we described in Sec. 5.4. The untriangulated LP produced fractional solutions for inference on the test data in several settings, which we rounded independently for each label. For networks with attractive potentials (AMNs), we used the LP in Sec. 7.2, which always produced integral solutions on test data.

8.4 Experiments

We tried out our framework on the *WebKB* dataset [Craven *et al.*, 1998], which is an instance of our hypertext example. The data set contains webpages from four different Computer Science departments: Cornell, Texas, Washington and Wisconsin. Each page has a label attribute, representing the type of webpage which is one of *course, faculty, student, project* or *other*. The data set is problematic in that the category *other* is a grab-bag of pages of many different types. The number of pages classified as *other* is quite large, so that a baseline algorithm that simply always selected *other* as the label would get an average accuracy of 75%. We could restrict attention to just the pages with the four other labels, but in a relational classification setting, the deleted webpages might be useful in terms of their interactions with other webpages. Hence, we compromised by eliminating all *other* pages with fewer than three outlinks, making the number of *other* pages commensurate with the other categories. The resulting category distribution is: course (237), faculty (148), other (332), research-project (82) and student (542). The number of remaining pages for each school are: Cornell (280), Texas (292), Washington (315) and Wisconsin (454). The number of links for each school are: Cornell (574), Texas (574), Washington (728) and Wisconsin (1614).

For each page, we have access to the entire html of the page and the links to other pages. Our goal is to collectively classify webpages into one of these five categories. In all of our experiments, we learn a model from three schools and test the performance of the learned model on the remaining school, thus evaluating the generalization performance of

the different models. We used $C \in [0.1, 10]$ and took the best setting for all models.

Unfortunately, we cannot directly compare our accuracy results with previous work because different papers use different subsets of the data and different training/test splits. However, we compare to standard text classifiers such as Naive Bayes, Logistic Regression, and Support Vector Machines, which have been demonstrated to be successful on this data set [Joachims, 1999].

8.4.1 Flat models

The simplest approach we tried predicts the categories based on just the text content on the webpage. The text of the webpage is represented using a set of binary attributes that indicate the presence of different words on the page. We found that stemming and feature selection did not provide much benefit and simply pruned words that appeared in fewer than three documents in each of the three schools in the training data. We also experimented with incorporating meta-data: words appearing in the title of the page, in anchors of links to the page and in the last header before a link to the page [Yang *et al.*, 2002]. Note that meta-data, although mostly originating from pages linking into the considered page, are easily incorporated as features, i.e. the resulting classification task is still flat feature-based classification. Our first experimental setup compares three well-known text classifiers — Naive Bayes, linear support vector machines (Svm), and logistic regression (Logistic) — using words and meta-words. The results, shown in Fig. 8.2, show that the two discriminative approaches outperform Naive Bayes. Logistic and Svm give very similar results. The average error over the 4 schools was reduced by around 4% by introducing the meta-data attributes.

Incorporating meta-data gives a significant improvement, but we can take additional advantage of the correlation in labels of related pages by classifying them collectively. We want to capture these correlations in our model and use them for transmitting information between linked pages to provide more accurate classification. We experimented with several relational models.

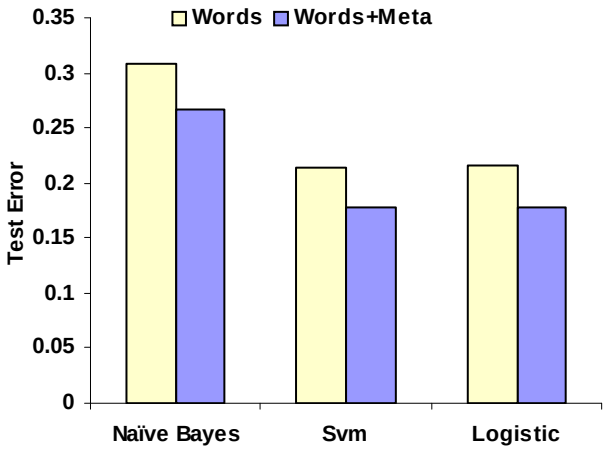


Figure 8.2: Comparison of Naive Bayes, Svm, and Logistic on WebKB, with and without meta-data features. (Only averages over the 4 schools are shown here.)

8.4.2 Link model

Our first model captures direct correlations between labels of linked pages. These correlations are very common in our data: courses and research projects almost never link to each other; faculty rarely link to each other; students have links to all categories but mostly courses. The Link model, shown in Fig. 8.1, captures this correlation through links: in addition to the local bag of words and meta-data attributes, we introduce a relational clique template over the labels of two pages that are linked.We train this model using maximum conditional likelihood (labels given the words and the links) and maximum margin.

We also compare to a directed graphical model to contrast discriminative and generative models of relational structure. The Exists-ML model is a (partially) generative model proposed by Getoor *et al.* [2001]. For each page, a logistic regression model predicts the page label given the words and meta-features. Then a simple generative model specifies a probability distribution over the existence of links between pages conditioned on both pages' labels. Concretely, we learn the probability of existence of a link between two pages given their labels. Note that this model does not require inference during learning. Maximum likelihood estimation (with regularization) of the generative component is closed

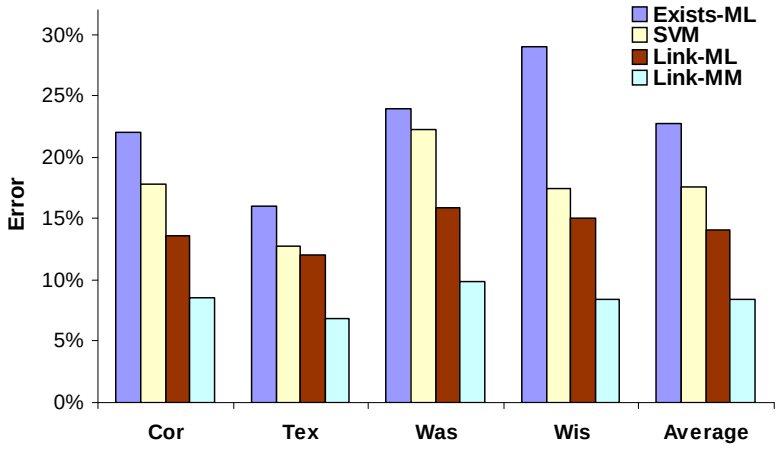


Figure 8.3: Comparison of flat versus collective classification on WebKB: SVM, Exists model with logistic regression and the Link model estimated using the maximum likelihood (ML) and the maximum margin (MM) criteria.

form given appropriate co-occurrence counts of linked pages' labels. However, the prediction phase is much more expensive, since the resulting graphical model includes edges not only for the existing hyperlinks, but also those that do not exist. Intuitively, observing the link structure directly correlates all page labels in a website, linked or not. By contrast, the Link model avoids this problem by only modeling the conditional distribution given the existing links.

Fig. 8.3 shows a gain in accuracy from SVMs to the Link model by using the correlations between labels of linked web pages. There is also very significant additional gain by using maximum margin training: the error rate of Link-MM is 40% lower than that of Link-ML, and 51% lower than multi-class SVMs. The Exists model doesn't perform very well in comparison. This can be attributed to the simplicity of the generative model and the difficulty of the resulting inference problem.

8.4.3 Cocite model

The second relational model uses the insight that a webpage often has internal structure that allows it to be broken up into *sections*. For example, a faculty webpage might have

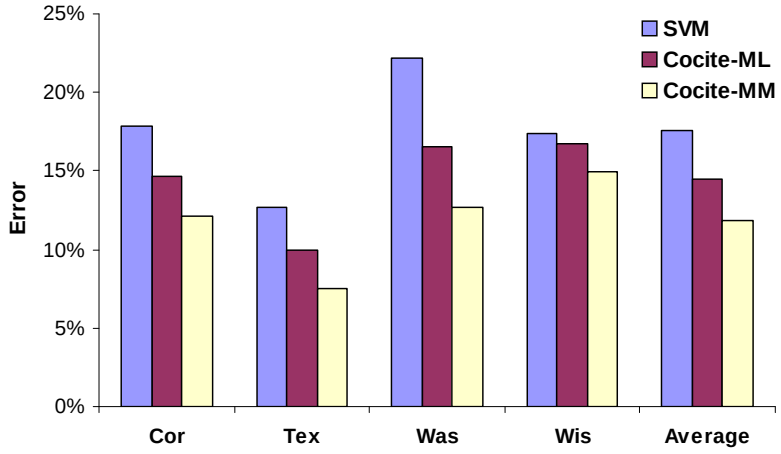


Figure 8.4: Comparison of Naive Bayes, Svm, and Logistic on WebKB, with and without meta-data features. (Only averages over the 4 schools are shown here.)

one section that discusses research, with a list of links to all of the projects of the faculty member, a second section might contain links to the courses taught by the faculty member, and a third to his advisees. We can view a section of a webpage as a fine-grained version of Kleinberg's hub [Kleinberg, 1999] (a page that contains a lot of links to pages of particular category). Intuitively, if two pages are *cocited*, or linked to from the same section, they are likely to be on similar topics. Note that we expect the correlation between the labels in this case to be positive, so we can use AMN-type potentials in the max-margin estimation. The Cocite model captures this type of correlation.

To take advantage of this trend, we need to enrich our schema by adding the attribute *Section* to Link to refer to the section number it appears in. We defined a section as a sequence of three or more links that have the same path to the root in the html parse tree. In the RMN, we have a relational clique template defined by:

```
SELECT doc1.Category, doc2.Category
FROM Doc doc1, Doc doc2, Link link1, Link link2
WHERE link1.From = link2.From and link1.Section = link2.Section and
        link1.To = doc1.Key and link2.To = doc2.Key and not doc1.Key = doc2.Key
```

We compared the performance of SVM, Cocite-ML and Cocite-MM. The results,

shown in Fig. 8.4, also demonstrate significant improvements of the relational models over the SVM. The improvement is present when testing on each of the schools. Again, maximum likelihood trained model Cocite-ML achieves a worse test error than maximum margin Cocite-MM model, which shows a 30% relative reduction in test error over SVM. We note that, in our experiments, the learned Cocite-MM weights never produced fractional solutions when used for inference, which suggests that the optimization successfully avoided problematic parameterizations of the network, even in the case of the non-optimal multi-class relaxation.

8.5 Related work

Our RMN representation is most closely related to the work on PRMs [Koller & Pfeffer, 1998]. Later work showed how to efficiently learn model parameters and structure (equivalent of clique selection in Markov networks) from data [Friedman *et al.*, 1999]. Getoor *et al.* [2002] propose several generative models of relational structure. Their approach easily captures the dependence of link existence on attributes of entities. However there are many patterns that we are difficult to model in PRMs, in particular those that involve several links at a time. We give some examples here.

One useful type of pattern type is a *similarity* template, where objects that share a certain graph-based property are more likely to have the same label. Consider, for example, a professor X and two other entities Y and Z. If X's webpage mentions Y and Z in the same context, it is likely that the X-Y relation and the Y-Z relation are of the same type; for example, if Y is Professor X's advisee, then probably so is Z. Our framework accommodates these patterns easily, by introducing pairwise cliques between the appropriate relation variables.

Another useful type of subgraph template involves *transitivity* patterns, where the presence of an A-B link and of a B-C link increases (or decreases) the likelihood of an A-C link. For example, students often assist in courses taught by their advisor. Note that this type of interaction cannot be accounted for just using pairwise cliques. By introducing cliques over triples of relations, we can capture such patterns as well. We can incorporate even more complicated patterns, but of course we are limited by the ability of belief propagation

to scale up as we introduce larger cliques and tighter loops in the Markov network.

We describe and exploit these patterns in our work on RMNs using maximum likelihood estimation [Taskar *et al.*, 2003b]. Attempts to model such pattern in PRMs run into the constraint that the probabilistic dependency graph (Bayesian network) must be a directed acyclic graph. For example, for the transitivity pattern, we might consider simply directing the correlation edges between link existence variables arbitrarily. However, it is not clear how to parameterize a link existence variable for a link that is involved in multiple triangles.

The structure of the relational graph has been used extensively to infer importance in scientific publications [Egghe & Rousseau, 1990] and hypertext [Kleinberg, 1999]. Several recent papers have proposed algorithms that use the link graph to aid classification. Chakrabarti *et al.* [1998] use system-predicted labels of linked documents to iteratively re-label each document in the test set, achieving a significant improvement compared to a baseline of using the text in each document alone. A similar approach was used by Neville and Jensen [2000] in a different domain. Slattery and Mitchell [2000] tried to identify directory (or hub) pages that commonly list pages of the same topic, and used these pages to improve classification of university webpages. However, none of these approaches provide a coherent model for the correlations between linked webpages, applying combinations of classifiers in a procedural way, with no formal justification.

8.6 Conclusion

In this chapter, we propose a new approach for classification in relational domains. Our approach provides a coherent foundation for the process of collective classification, where we want to classify multiple entities, exploiting the interactions between their labels. We have shown that we can exploit a very rich set of relational patterns in classification, significantly improving the classification accuracy over standard flat classification.

In some cases, we can incorporate relational features into standard flat classification. For example, when classifying papers into topics, it is possible to simply view the presence of particular citations as atomic features. However, this approach is limited in cases where some or even all of the relational features that occur in the test data are not observed in the training data. In our WebKB example, there is no overlap between the webpages in the

different schools, so we cannot learn anything from the training data about the significance of a hyperlink to/from a particular webpage in the test data. Incorporating basic features (e.g., words) from the related entities can aid in classification, but cannot exploit the strong correlation between the *labels* of related entities that RMNs capture.

Hypertext is the most easily available source of structured data, however, RMNs are generally applicable to any relational domain. The results in this chapter represent only a subset of the domains we have worked on (see [Taskar *et al.*, 2003b]). In particular, social networks provide extensive information about interactions among people and organizations. RMNs offer a principled method for learning to predict communities of and hierarchical structure between people and organizations based on both the local attributes and the patterns of static and dynamic interaction. Given the wealth of possible patterns, it is particularly interesting to explore the problem of inducing them automatically.

Part III

Broader applications: parsing, matching, clustering

Chapter 9

Context free grammars

We present a novel discriminative approach to parsing using structured max-margin criterion based on the decomposition properties of context free grammars. We show that this framework allows high-accuracy parsing in cubic time by exploiting novel kinds of lexical information. Our models can condition on arbitrary features of input sentences, thus incorporating an important kind of lexical information not usually used by conventional parsers. We show experimental evidence of the model's improved performance over a natural baseline model and a lexicalized probabilistic context-free grammar.

9.1 Context free grammar model

CFGs are one of the primary formalisms for capturing the recursive structure of syntactic constructions, although many others have also been proposed [Manning & Schütze, 1999]. For clarity of presentation, we restrict our grammars to be in Chomsky normal form as in Sec. 3.4. The non-terminal symbols correspond to syntactic categories such as noun phrase (NP) or verbal phrase (VP). The terminal symbols are usually words of the sentence. However, in the discriminative framework that we adopt, we are not concerned with defining a distribution over sequences of words (language model). Instead, we *condition* on the words in a sentence to produce a model of the syntactic structure. Terminal symbols for our purposes are part-of-speech tags like nouns (NN), verbs (VBD), determiners (DT). For example, Fig. 9.1(a) shows a parse tree for the sentence *The screen was a sea of*

red. The set of symbols we use is based on the Penn Treebank [Marcus *et al.*, 1993]. The non-terminal symbols with bars (for example, $\overline{\mathrm{DT}}, \overline{\mathrm{NN}}, \overline{\mathrm{VBD}}$) are added to conform to the CNF restrictions. For convenience, we repeat our definition of a CFG from Sec. 3.4 here:

Definition 9.1.1 (CFG) *A CFG $\mathcal{G}$ consists of:*

- *A set of non-terminal symbols, $\mathcal{N}$*
- *A designated set of start symbols, $\mathcal{N}_S \subseteq \mathcal{N}$*
- *A set of terminal symbols, $\mathcal{T}$*
- *A set of productions, $\mathcal{P} = \{\mathcal{P}_B, \mathcal{P}_U\}$, divided into*
  - *Binary productions, $\mathcal{P}_B = \{A \to B\ C : A, B, C \in \mathcal{N}\}$ and*
  - *Unary productions, $\mathcal{P}_U = \{A \to D : A \in \mathcal{N}, D \in \mathcal{T}\}$.*

A CFG defines a set of valid parse trees in a natural manner:

Definition 9.1.2 (CFG tree) *A* CFG tree *is a labeled directed tree, where the set of valid labels of the internal nodes other than the root is $\mathcal{N}$ and the set of valid labels for the leaves is $\mathcal{T}$. The root's label set is $\mathcal{N}_S$. Additionally, each pre-leaf node has a single child and this pair of nodes can be labeled as A and D, respectively, if and only if there is a unary production $A \to D \in \mathcal{P}_U$. All other internal nodes have two children, left and right, and this triple of nodes can be labeled as A, B and C, respectively, if and only if there is a binary production $A \to B\ C \in \mathcal{P}_B$.*

In general, there are exponentially many parse trees that produce a sentence of length n.

This tree representation seems quite different from the graphical models we have been considering thus far. However, we can use an equivalent representation that essentially encodes a tree as an assignment to a set of appropriate variables. For each span starting with s and an ending with e, we introduce a variable $Y_{s,e}$ taking values in $\mathcal{N} \cup \perp$ to represent the label of the subtree that exactly covers, or *dominates*, the words of the sentence from s to e. The value $\perp$ is assigned if no subtree dominates the span. Indices s and e refer to positions between words, rather than to words themselves, hence $0 \leq s < e \leq n$ for a sentence of length n. The "top" symbol $Y_{0,n}$ is constrained to be in $\mathcal{N}_S$, since it represents

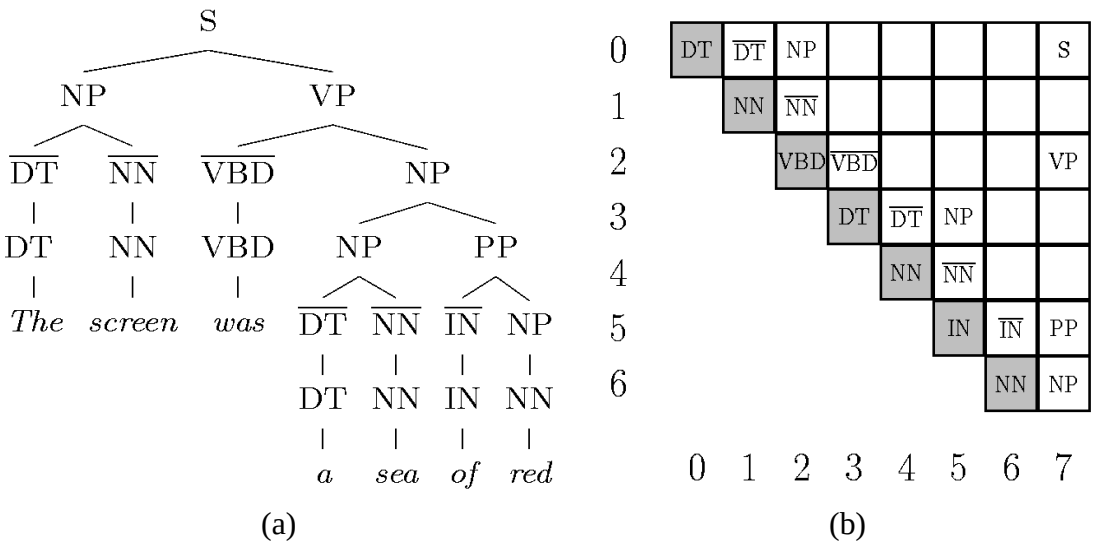


Figure 9.1: Two representations of a binary parse tree: (a) nested tree structure, and (b) grid of labeled spans. The row and column number are the beginning and end of the span, respectively. Empty squares correspond to non-constituent spans. The gray squares on the diagonal represent part-of-speech tags.

the starting symbol of the sentence. We also introduce variables $Y_{s,s}$ taking values in $\mathcal{T}$ to represent the terminal symbol (part-of-speech) between s and $s+1$. If $Y_{s,e} \neq \perp$, it is often called a *constituent*. Fig. 9.1(b) shows the representation of the tree in Fig. 9.1(a) as a grid where each square corresponds to $Y_{s,e}$. The row and column number in the grid correspond to the beginning and end of the span, respectively. Empty squares correspond to $\perp$ values. The gray squares on the diagonal represent the terminal variables. For example, the figure shows $Y_{0,0} = \text{DT}$, $Y_{6,6} = \text{NN}$, $Y_{3,5} = \text{NP}$ and $Y_{1,4} = \perp$.

While any parse tree corresponds to an assignment to this set of variables in a straightforward manner, the converse is not true: there are assignments that do not correspond to valid parse trees. In order to characterize the set of valid assignments $\mathcal{Y}$, consider the

constraints that hold for a valid assignment $\mathbf{y}$:

$$\mathbb{1}(y_{s,e} = A) = \sum_{\substack{A \to B\ C \in \mathcal{P}_B \\ s<m<e}} \mathbb{1}(\mathbf{y}_{s,m,e} = (A, B, C)), \quad 0 \leq s < e \leq n,\ \forall A \in \mathcal{N}; \tag{9.1}$$

$$\mathbb{1}(y_{s,e} = A) = \sum_{\substack{B \to A\ C \in \mathcal{P}_B \\ 0 \leq s' < s}} \mathbb{1}(\mathbf{y}_{s',s,e} = (B, A, C)) + \sum_{\substack{B \to C\ A \in \mathcal{P}_B \\ e<e' \leq n}} \mathbb{1}(\mathbf{y}_{s,e,e'} = (B, C, A)), \quad 0 \leq s < e \leq n,\ \forall A \in \mathcal{N}; \tag{9.2}$$

$$\mathbb{1}(y_{s,s+1} = A) = \sum_{A \to D \in \mathcal{P}_U} \mathbb{1}(\mathbf{y}_{s,s,s+1} = (A, D)), \quad 0 \leq s < n,\ \forall A \in \mathcal{N}; \tag{9.3}$$

$$\mathbb{1}(y_{s,s} = D) = \sum_{A \to D \in \mathcal{P}_U} \mathbb{1}(\mathbf{y}_{s,s,s+1} = (A, D)), \quad 0 \leq s < n,\ \forall D \in \mathcal{T}. \tag{9.4}$$

The notation $\mathbf{y}_{s,m,e} = (A, B, C)$ abbreviates $y_{s,e} = A \wedge y_{s,m} = B \wedge y_{m,e} = C$ and $\mathbf{y}_{s,s,s+1} = (A, D)$ abbreviates $y_{s,s+1} = A \wedge y_{s,s} = D$. The first set of constraints (9.1) holds because if the span from s to e is dominated by a subtree starting with A (that is, $y_{s,e} = A$), then there must be a unique production starting with A and some split point m, $s < m < e$, that produces that subtree. Conversely, if $y_{s,e} \neq A$, no productions start with A and cover s to e. The second set of constraints (9.2) holds because that if the span from s to e is dominated by a subtree starting with A ($y_{s,e} = A$), then there must be a (unique) production that generated it: either starting before s or after e. Similarly, the third and fourth set of constraints (9.3 and 9.4) hold since the terminals are generated using valid unary productions. We denote the set of assignments $\mathbf{y}$ satisfying (9.1-9.4) as $\mathcal{Y}$. In fact the converse is true as well:

Theorem 9.1.3 *If* $\mathbf{y} \in \mathcal{Y}$*, then* $\mathbf{y}$ *represents a valid CFG tree.*

Proof sketch: It is straightforward to construct a parse tree from $\mathbf{y} \in \mathcal{Y}$ in a top-down manner. Starting from the root symbol, $y_{0,n}$, the first set of constraints (9.1) ensures that a unique production spans 0 to n, say splitting at m and specifying the values for $y_{0,m}$ and $y_{m,n}$. The second set of constraints (9.2) ensures that all other spans $y_{0,m'}$ and $y_{m',n}$, for $m' \neq m$ are labeled by $\perp$. Recursing on the two subtrees, $y_{0,m}$ and $y_{m,n}$, will produce the rest of the tree down to the pre-terminals. The last two sets of constraints (9.3 and 9.4)

ensure that the terminals are generated by an appropriate unary productions from the pre-terminals.

9.2 Context free parsing

A standard approach to parsing is to use a CFG to define a probability distribution over parse trees. This can be done simply by assigning a probability to each production and making sure that the sum of probabilities of all productions starting with each symbol is 1:

$$\sum_{B,C:A\to B\ C\in\mathcal{P}_B} P(A \to B\ C) = 1, \qquad \sum_{D:A\to D\in\mathcal{P}_U} P(A \to D) = 1, \qquad \forall A \in \mathcal{N}.$$

The probability of a tree is simply the product of probabilities of the productions used in the tree. More generally, a weighted CFG assigns a score to each production (this score may depend on the position of the production s, m, e) such that the total score of a tree is the sum of the score of all the productions used:

$$S(\mathbf{y}) = \sum_{0\le s<m<e\le n} S_{s,m,e}(\mathbf{y}_{s,m,e}) + \sum_{0\le s<n} S_{s,s+1}(\mathbf{y}_{s,s+1}),$$

where $S_{s,m,e}(\mathbf{y}_{s,m,e}) = 0$ if $(y_{s,e} = \perp \vee y_{s,m} = \perp \vee y_{m,e} = \perp)$. If the production scores are production log probabilities, then the tree score is the tree log probability. However, weighted CFGs do not have the local normalization constraints Eq. (9.5).

We can use a Viterbi-style dynamic programming algorithm called CKY to compute the highest score parse tree in $\mathcal{O}(|\mathcal{P}|n^3)$ time [Younger, 1967; Manning & Schütze, 1999]. The algorithm computes the highest score of any subtree starting with a symbol over each span $0 \le s < e \le n$ recursively:

$$S^*_{s,s+1}(A) = \max_{A\to D\in\mathcal{P}_U} S_{s,s+1}(A, D), \quad 0 \le s < n,\ \forall A \in \mathcal{N}; \tag{9.5}$$
$$S^*_{s,e}(A) = \max_{\substack{A\to B\ C\in\mathcal{P}_B \\ s<m<e}} S_{s,m,e}(A, B, C) + S^*_{s,m}(B) + S^*_{m,e}(C), \quad 0 \le s < e \le n,\ \forall A \in \mathcal{N}.$$

The highest scoring tree has score $\max_{A\in\mathcal{N}_S} S^*_{0,n}(A)$. Using the $\arg\max$'s of the $\max$'s in

the computation of S^*, we can back-trace the highest scoring tree itself. We assume that score ties are broken in a predetermined way, say according to some lexicographic order of the symbols.

9.3 Discriminative parsing models

We cast parsing as a structured classification task, where we want to learn a function $h : \mathcal{X} \mapsto \mathcal{Y}$, where $\mathcal{X}$ is a set of sentences, and $\mathcal{Y}$ is a set of valid parse trees according to a fixed CFG grammar.

The functions we consider take the following linear discriminant form:

$$h_{\mathbf{w}}(\mathbf{x}) = \arg\max_{\mathbf{y}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}),$$

where $\mathbf{w} \in \mathbb{R}^d$ and $\mathbf{f}$ is a basis function representation of a sentence and parse tree pair $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$. We assume that the basis functions decompose with the CFG structure:

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{0 \le s \le e \le n} \mathbf{f}(\mathbf{x}_{s,e}, y_{s,e}) + \sum_{0 \le s \le m < e \le n} \mathbf{f}(\mathbf{x}_{s,m,e}, \mathbf{y}_{s,m,e}),$$

where n is the length of the sentence $\mathbf{x}$ and $\mathbf{x}_{s,e}$ and $\mathbf{x}_{s,m,e}$ are the relevant subsets of the sentence the basis functions depend on. To simplify notation, we introduce the set of indices, $\mathcal{C}$, which includes both spans and span triplets:

$$\mathcal{C} = \{(s, m) : 0 \le s \le e \le n\} \ \cup \ \{(s, m, e) : 0 \le s \le m < e \le n\}.$$

Hence, $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{c \in \mathcal{C}} \mathbf{f}(\mathbf{x}_c, y_c)$.

Note that this class of discriminants includes PCFG models, where the basis functions consist of the counts of the productions used in the parse, and the parameters $\mathbf{w}$ are the log-probabilities of those productions. For example, $\mathbf{f}$ could include functions which identify the production used together with features of the words at positions s, m, e, and neighboring positions in the sentence $\mathbf{x}$ (e.g. $f(\mathbf{x}_{s,m,e}, \mathbf{y}_{s,m,e}) = \mathbb{1}(\mathbf{y}_{s,m,e} = \mathrm{S}, \mathrm{NP}, \mathrm{VP}) \wedge m^{th} word(\mathbf{x}) = was$)). We could also include functions that identify the label of the span

from s to e together with features of the word (e.g. $f(\mathbf{x}_{s,m}, y_{s,m}) = \mathbb{1}(y_{s,m} = \text{NP}) \wedge s^{th}word(\mathbf{x}) = the)$).

9.3.1 Maximum likelihood estimation

The traditional method of estimating the parameters of PCFGs assumes a generative model that defines $P(\mathbf{x}, \mathbf{y})$ by assigning normalized probabilities to CFG productions. We then maximize the joint log-likelihood $\sum_i \log P(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ (with some regularization). We compare to such a generative grammar of Collins [1999] in our experiments.

A alternative probabilistic approach is to estimate the parameters discriminatively by maximizing *conditional* log-likelihood. For example, the maximum entropy approach [Johnson, 2001] defines a conditional log-linear model:

$$P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z_{\mathbf{w}}(\mathbf{x})} \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})\},$$

where $Z_{\mathbf{w}}(\mathbf{x}) = \sum_{\mathbf{y}} \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})\}$, and maximizes the conditional log-likelihood of the sample, $\sum_i \log P(\mathbf{y}^{(i)} \mid \mathbf{y}^{(i)})$, (with some regularization). The same assumption that the basis functions decompose as sums of local functions over spans and productions is typically made in such models. Hence, as in Markov networks, the gradient depends on the expectations of the basis functions, which can be computed in $\mathcal{O}(|\mathcal{P}|n^3)$ time by dynamic programming algorithm called inside-outside, which is similar to the CKY algorithm. However, computing the expectations over trees is actually more expensive in practice than finding the best tree for several reasons. CKY works entirely in the log-space, while inside-outside needs to compute actual probabilities. Branch-and-prune techniques, which save a lot of useless computation, are only applicable in CKY.

A typical method for finding the parameters is to use Conjugate Gradients or L-BFGS methods [Nocedal & Wright, 1999; Boyd & Vandenberghe, 2004], which repeatedly compute these expectations to calculate the gradient. Clark and Curran [2004] report experiments involving 479 iterations of training for one model, and 1550 iterations for another using similar methods.

9.3.2 Maximum margin estimation

We assume that loss function also decomposes with the CFG structure:

$$\ell(\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}}) = \sum_{0 \le s \le e \le n} \ell(\mathbf{x}_{s,e}, y_{s,e}, \hat{y}_{s,e}) + \sum_{0 \le s \le m < e \le n} \ell(\mathbf{x}_{s,m,e}, \mathbf{y}_{s,m,e}, \hat{\mathbf{y}}_{s,m,e}) = \sum_{c \in \mathcal{C}} \ell(\mathbf{x}_c, \mathbf{y}_c, \hat{\mathbf{y}}_c).$$

One approach would be to define $\ell(\mathbf{x}_{s,e}, y_{s,e}, \hat{y}_{s,e}) = \mathbb{1}(y_{s,e} \neq \hat{y}_{s,e})$. This would lead to $\ell(\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}})$ tracking the number of "constituent errors" in $\hat{\mathbf{y}}$. Another, more strict definition would be to define $\ell(\mathbf{x}_{s,m,e}, \mathbf{y}_{s,m,e}, \hat{\mathbf{y}}_{s,m,e}) = \mathbb{1}(\mathbf{y}_{s,m,e} \neq \hat{\mathbf{y}}_{s,m,e})$. This definition would lead to $\ell(\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}})$ being the number of productions in $\hat{\mathbf{y}}$ which are not seen in $\mathbf{y}$. The constituent loss function does not exactly correspond to the standard scoring metrics, such as F_1 or crossing brackets, but shares the sensitivity to the number of differences between trees. We have not thoroughly investigated the exact interplay between the various loss choices and the various parsing metrics. We used the constituent loss in our experiments.

As in the max-margin estimation for Markov networks, we can formulate an exponential size QP:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_i \xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y}) \ge \ell_i(\mathbf{y}) - \xi_i \quad \forall i, \mathbf{y}, \end{aligned} \tag{9.6}$$

where $\Delta \mathbf{f}_i(\mathbf{y}) = \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y})$, and $\ell_i(\mathbf{y}) = \ell(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{y})$.

The dual of Eq. (9.6) (after normalizing by C) is given by:

$$\begin{aligned} \max \quad & \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y}) \ell_i(\mathbf{y}) - \frac{1}{2} C \left|\left| \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y}) \Delta \mathbf{f}_i(\mathbf{y}) \right|\right|^2 \\ \text{s.t.} \quad & \sum_{\mathbf{y}} \alpha_i(\mathbf{y}) = 1, \quad \forall i; \quad \alpha_i(\mathbf{y}) \ge 0, \quad \forall i, \mathbf{y}. \end{aligned} \tag{9.7}$$

Both of the above formulations are exponential (in the number of variables or constraints) in the lengths (n_i's) of the sentences. But we can exploit the context-free structure of the basis functions and the loss to define a polynomial-size dual formulation in terms of

marginal variables $\mu_i(\mathbf{y})$:

$$\mu_{i,s,e}(A) \equiv \sum_{\mathbf{y}:y_{s,e}=A} \alpha_i(\mathbf{y}), \quad 0 \leq s \leq e \leq n, \ \forall A \in \mathcal{N};$$
$$\mu_{i,s,s}(D) \equiv \sum_{\mathbf{y}:y_{s,s}=D} \alpha_i(\mathbf{y}), \quad 0 \leq s \leq e \leq n, \ \forall D \in \mathcal{T};$$
$$\mu_{i,s,m,e}(A,B,C) \equiv \sum_{\mathbf{y}:y_{s,m,e}=(A,B,C)} \alpha_i(\mathbf{y}), \quad 0 \leq s < m < e \leq n, \ \forall A \to B\ C \in \mathcal{P}_B,$$
$$\mu_{i,s,s,s+1}(A,D) \equiv \sum_{\mathbf{y}:y_{s,s,s+1}=(A,D)} \alpha_i(\mathbf{y}), \quad 0 \leq s < n, \ \forall A \to D \in \mathcal{P}_U.$$

There are $\mathcal{O}(|\mathcal{P}_B|n_i^3 + |\mathcal{P}_U|n_i)$ such variables for each sentence of length n_i, instead of exponentially many α_i variables. We can now express the objective function in terms of the marginals. Using these variables, the first set of terms in the objective becomes:

$$\sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})\ell_i(\mathbf{y}) = \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y}) \sum_{c \in \mathcal{C}^{(i)}} \ell_{i,c}(\mathbf{y}_c) = \sum_{i,c \in \mathcal{C}^{(i)},\mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c)\ell_{i,c}(\mathbf{y}_c).$$

Similarly, the second set of terms (inside the 2-norm) becomes:

$$\sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y}) = \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y}) \sum_{c \in \mathcal{C}^{(i)}} \Delta\mathbf{f}_{i,c}(\mathbf{y}_c) = \sum_{i,c \in \mathcal{C}^{(i)},\mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c)\Delta\mathbf{f}_{i,c}(\mathbf{y}_c).$$

As in M^{3}Ns, we must characterize the set of marginals μ that corresponds to valid α. The constraints on μ are essentially based on the those that define $\mathcal{Y}$ in (9.1-9.4). In addition, we require that the marginals over the root nodes, $\mu_{i,0,n_i}(y_{0,n_i})$, sums to 1 over the possible start symbols $\mathcal{N}_S$.

Putting the pieces together, the factored dual is:

$$
\begin{aligned}
\max \quad & \sum_{i,c\in\mathcal{C}^{(i)}} \mu_{i,c}(\mathbf{y}_c)\ell_{i,c}(\mathbf{y}_c) + C\left\|\sum_{i,c\in\mathcal{C}^{(i)}} \mu_{i,c}(\mathbf{y}_c)\Delta\mathbf{f}_{i,c}(\mathbf{y}_c)\right\|^2 \qquad (9.8)\\
\text{s.t.} \quad & \sum_{A\in\mathcal{N}_S} \mu_{i,0,n_i}(A) = 1, \quad \forall i; \qquad \mu_{i,c}(\mathbf{y}_c) \geq 0;\ \forall i,\ \forall c \in \mathcal{C}^{(i)};\\
& \mu_{i,s,e}(A) = \sum_{\substack{A\to B\ C\in\mathcal{P}_B\\ s<m<e}} \mu_{i,s,m,e}(A,B,C), \quad \forall i,\ 0\leq s<e\leq n_i,\ \forall A\in\mathcal{N};\\
& \mu_{i,s,e}(A) = \sum_{\substack{B\to A\ C\in\mathcal{P}_B\\ 0\leq s'<s}} \mu_{i,s',s,e}(B,A,C)\\
& \qquad + \sum_{\substack{B\to C\ A\in\mathcal{P}_B\\ e<e'\leq n_i}} \mu_{i,s,e,e'}(B,C,A),\ \forall i, 0\leq s<e\leq n_i, \forall A\in\mathcal{N};\\
& \mu_{i,s,s+1}(A) = \sum_{A\to D\in\mathcal{P}_U} \mu_{i,s,s,s+1}(A,D), \quad \forall i,\ 0\leq s<n_i,\ \forall A\in\mathcal{N};\\
& \mu_{i,s,s}(D) = \sum_{A\to D\in\mathcal{P}_U} \mu_{i,s,s,s+1}(A,D), \quad \forall i,\ 0\leq s<n_i,\ \forall D\in\mathcal{T}.
\end{aligned}
$$

The constraints on μ is necessary, since they must correspond to marginals of a distribution over trees. They are also sufficient:

Theorem 9.3.1 *A set of marginals $\mu_i(\mathbf{y})$ satisfying the constraints in Eq. (9.8) corresponds to a valid distribution over the legal parse trees $\mathbf{y} \in \mathcal{Y}^{(i)}$. A consistent distribution $\alpha_i(\mathbf{y})$ is given by*

$$\alpha_i(\mathbf{y}) = \mu_{i,0,n_i}(y_{0,n_i}) \prod_{0\leq s\leq m<e\leq n_i} \frac{\mu_{i,s,m,e}(\mathbf{y}_{s,m,e})}{\mu_{i,s,e}(y_{s,e})},$$

where $0/0 = 0$ by convention.

Proof sketch: The proof follows from inside-outside probability relations [Manning & Schütze, 1999]. The first term is a valid distribution of starting symbols. Each $\frac{\mu_{i,s,m,e}(\mathbf{y}_{s,m,e})}{\mu_{i,s,e}(y_{s,e})}$ term for $m > s$ corresponds to a conditional distribution over binary productions $(y_{s,e} \to y_{s,m}\ y_{m,e})$ that are guaranteed to sum to 1 over split points m and possible productions. Similarly, each $\frac{\mu_{i,s,s,s+1}(\mathbf{y}_{s,s,s+1})}{\mu_{i,s,s+1}(y_{s,s+1})}$ term for corresponds to a conditional distribution over

unary productions $(y_{s,s+1} \rightarrow y_{s,s})$ that are guaranteed to sum to 1 over possible productions. Hence, we have defined a kind of PCFG (where production probabilities depend on the location of the symbol), which induces a valid distribution α_i over trees. It straightforward to verify that this distribution has marginals μ_i.

9.4 Structured SMO for CFGs

We trained our max-margin models using the Structured SMO algorithm with block-coordinate descent adopted from graphical models (see Sec. 6.1). The CKY algorithm computes similar max-marginals in the course of computing the best tree as does Viterbi in Markov networks.

$$\widehat{v}_{i,c}(\mathbf{y}_c) = \max_{\mathbf{y} \sim \mathbf{y}_c} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})], \qquad \widehat{\alpha}_{i,c}(\mathbf{y}_c) = \max_{\mathbf{y} \sim \mathbf{y}_c} \alpha_i(\mathbf{y}).$$

We also define $\widehat{v}_{i,c}(\overline{\mathbf{y}_c}) = \max_{\mathbf{y}'_c \neq \mathbf{y}_c} \widehat{v}_{i,c}(\mathbf{y}'_c) = \max_{\mathbf{y} \not\sim \mathbf{y}_c} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$. Note that we do not explicitly represent $\alpha_i(\mathbf{y})$, but we can reconstruct the maximum-entropy one from the marginals μ_i as in Theorem 9.3.1.

We again express the KKT conditions in terms of the max-marginals for each span and span triple $c \in \mathcal{C}^{(i)}$ and its values $\mathbf{y}_c$:

$$\widehat{\alpha}_{i,c}(\mathbf{y}_c) = 0 \Rightarrow \widehat{v}_{i,c}(\mathbf{y}_c) \leq \widehat{v}_{i,c}(\overline{\mathbf{y}_c}); \qquad \widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0 \Rightarrow \widehat{v}_{i,c}(\mathbf{y}_c) \geq \widehat{v}_{i,c}(\overline{\mathbf{y}_c}). \tag{9.9}$$

The algorithm cycles through the training sentences, runs CKY to compute the max-marginals and performs an SMO update on the violated constraints. We typically find that 20-40 iterations through the data are sufficient for convergence in terms of the objective function improvements.

9.5 Experiments

We used the Penn English Treebank for all of our experiments. We report results here for each model and setting trained and tested on only the sentences of length ≤ 15 words. Aside

from the length restriction, we used the standard splits: sections 2-21 for training (9753 sentences), 22 for development (603 sentences), and 23 for final testing (421 sentences).

As a baseline, we trained a CNF transformation of the unlexicalized model of Klein and Manning [2003] on this data. The resulting grammar had 3975 non-terminal symbols and contained two kinds of productions: binary non-terminal rewrites and tag-word rewrites. Unary rewrites were compiled into a single compound symbol, so for example a subject-gapped sentence would have label like S+VP. These symbols were expanded back into their source unary chain before parses were evaluated. The scores for the binary rewrites were estimated using unsmoothed relative frequency estimators. The tagging rewrites were estimated with a smoothed model of $P(w|t)$, also using the model from Klein and Manning [2003]. In particular, Table 9.2 shows the performance of this model (GENERATIVE): 87.99 F_1 on the test set.

For the BASIC max-margin model, we used exactly the same set of allowed rewrites (and therefore the same set of candidate parses) as in the generative case, but estimated their weights using the max-margin formulation with a loss that counts the number of wrong spans. Tag-word production weights were fixed to be the log of the generative $P(w|t)$ model. That is, the only change between GENERATIVE and BASIC is the use of the discriminative maximum-margin criterion in place of the generative maximum likelihood one for learning production weights. This change alone results in a small improvement (88.20 vs. 87.99 F_1).

On top of the basic model, we first added lexical features of each span; this gave a LEXICAL model. For a span $\langle s, e\rangle$ of a sentence x, the base lexical features were:

- x_s, the first word in the span
- x_{s-1}, the preceding adjacent word
- x_{e-1}, the last word in the span
- x_e, the following adjacent word
- $\langle x_{s-1}, x_s\rangle$
- $\langle x_{e-1}, x_e\rangle$
- x_{s+1} for spans of length 3

Model	P	R	F_1
GENERATIVE	87.70	88.06	87.88
BASIC	87.51	88.44	87.98
LEXICAL	88.15	88.62	88.39
LEXICAL+AUX	89.74	90.22	89.98

Table 9.1: Development set results of the various models when trained and tested on Penn treebank sentences of length ≤ 15.

Model	P	R	F_1
GENERATIVE	88.25	87.73	87.99
BASIC	88.08	88.31	88.20
LEXICAL	88.55	88.34	88.44
LEXICAL+AUX	89.14	**89.10**	**89.12**
COLLINS 99	**89.18**	88.20	88.69

Table 9.2: Test set results of the various models when trained and tested on Penn treebank sentences of length ≤ 15.

These base features were conjoined with the span length for spans of length 3 and below, since short spans have highly distinct behaviors (see the examples below). The features are lexical in the sense than they allow specific words and word pairs to influence the parse scores, but are distinct from traditional lexical features in several ways. First, there is no notion of headword here, nor is there any modeling of word-to-word attachment. Rather, these features pick up on lexical trends in constituent boundaries, for example the trend that in the sentence *The screen was a sea of red.*, the (length 2) span between the word *was* and the word *of* is unlikely to be a constituent. These non-head lexical features capture a potentially very different source of constraint on tree structures than head-argument pairs, one having to do more with linear syntactic preferences than lexical selection. Regardless of the relative merit of the two kinds of information, one clear advantage of the present approach is that inference in the resulting model remains cubic (as opposed to $\mathcal{O}(n^5)$), since the dynamic program need not track items with distinguished headwords. With the addition of these features, the accuracy moved past the generative baseline, to 88.44.

As a concrete (and particularly clean) example of how these features can sway a decision, consider the sentence *The Egyptian president said he would visit Libya today to resume the talks*. The generative model incorrectly considers *Libya today* to be a base NP. However, this analysis is counter to the trend of *today* to be a one-word constituent. Two features relevant to this trend are: (CONSTITUENT $\wedge$ first-word = *today* $\wedge$ length = 1) and (CONSTITUENT $\wedge$ last-word = *today* $\wedge$ length = 1). These features represent the preference of the word *today* for being the first and last word in constituent spans of length 1.[1] In the LEXICAL model, these features have quite large positive weights: 0.62 each. As a result, this model makes this parse decision correctly.

Another kind of feature that can usefully be incorporated into the classification process is the output of other, auxiliary classifiers. For this kind of feature, one must take care that its reliability on the training not be vastly greater than its reliability on the test set. Otherwise, its weight will be artificially (and detrimentally) high. To ensure that such features are as noisy on the training data as the test data, we split the training into two folds. We then trained the auxiliary classifiers on each fold, and using their predictions as features on the other fold. The auxiliary classifiers were then retrained on the entire training set, and their predictions used as features on the development and test sets.

We used two such auxiliary classifiers, giving a prediction feature for each span (these classifiers predicted only the presence or absence of a bracket over that span, not bracket labels). The first feature was the prediction of the generative baseline; this feature added little information, but made the learning phase faster. The second feature was the output of a flat classifier which was trained to predict whether single spans, in isolation, were constituents or not, based on a bundle of features including the list above, but also the following: the preceding, first, last, and following tag in the span, pairs of tags such as preceding-first, last-following, preceding-following, first-last, and the entire tag sequence.

Tag features on the test sets were taken from a pretagging of the sentence by the tagger described in [Toutanova *et al.*, 2003].While the flat classifier alone was quite poor (P 78.77 / R 63.94 / F_1 70.58), the resulting max-margin model (LEXICAL+AUX) scored 89.12 F_1. To situate these numbers with respect to other models, the parser in [Collins, 1999],which

[1] In this length 1 case, these are the same feature. Note also that the features are conjoined with only one generic label class "constituent" rather than specific constituent types.

is generative, lexicalized, and intricately smoothed scores 88.69 over the same train/test configuration.

9.6 Related work

A number of recent papers have considered discriminative approaches for natural language parsing [Johnson *et al.*, 1999; Collins, 2000; Johnson, 2001; Geman & Johnson, 2002; Miyao & Tsujii, 2002; Clark & Curran, 2004; Kaplan *et al.*, 2004; Collins, 2004]. Broadly speaking, these approaches fall into two categories, *reranking* and *dynamic programming* approaches. In reranking methods [Johnson *et al.*, 1999; Collins, 2000; Shen *et al.*, 2003], an initial parser is used to generate a number of candidate parses. A discriminative model is then used to choose between these candidates. In dynamic programming methods, a large number of candidate parse trees are represented compactly in a parse tree forest or chart. Given sufficiently "local" features, the decoding and parameter estimation problems can be solved using dynamic programming algorithms. For example, several approaches [Johnson, 2001; Geman & Johnson, 2002; Miyao & Tsujii, 2002; Clark & Curran, 2004; Kaplan *et al.*, 2004] are based on conditional log-linear (maximum entropy) models, where variants of the inside-outside algorithm can be used to efficiently calculate gradients of the log-likelihood function, despite the exponential number of trees represented by the parse forest.

The method we presented has several compelling advantages. Unlike reranking methods, which consider only a pre-pruned selection of "good" parses, our method is an end-to-end discriminative model over the full space of parses. This distinction can be very significant, as the set of n-best parses often does not contain the true parse. For example, in the work of Collins [2000], 41% of the correct parses were not in the candidate pool of $\sim$30-best parses. Unlike previous dynamic programming approaches, which were based on maximum entropy estimation, our method incorporates an articulated loss function which penalizes larger tree discrepancies more severely than smaller ones.

Moreover, the structured SMO we use requires only the calculation of Viterbi trees, rather than expectations over all trees (for example using the inside-outside algorithm).

This allows a range of optimizations that prune the space of parses (without making approximations) not possible for maximum likelihood approaches which must extract basis function expectations from the entire set of parses. In our experiments, 20-40 iterations were generally required for convergence (except the BASIC model, which took about 100 iterations.)

9.7 Conclusion

We have presented a maximum-margin approach to parsing, which allows a discriminative SVM-like objective to be applied to the parsing problem. Our framework permits the use of a rich variety of input features, while still decomposing in a way that exploits the shared substructure of parse trees in the standard way.

It is worth considering the cost of this kind of method. At training time, discriminative methods are inherently expensive, since they all involve iteratively checking current model performance on the training set, which means parsing the training set (usually many times). Generative approaches are vastly cheaper to train, since they must only collect counts from the training set.

On the other hand, the max-margin approach does have the potential to incorporate many new kinds of features over the input, and the current feature set allows limited lexicalization in cubic time, unlike other lexicalized models (including the Collins model which it outperforms in the present limited experiments). This trade-off between the complexity, accuracy and efficiency of a parsing model is an important area of future research.

Chapter 10

Matchings

We address the problem of learning to match: given a set of input graphs and corresponding matchings, find a parameterized edge scoring function such that the correct matchings have the highest score. Bipartite matchings are used in many fields, for example, to find marker correspondences in vision problems, to map words of a sentence in one language to another, to identify functional genetic analogues in different organisms. We have shown a compact max-margin formulation for bipartite matchings in Ch. 4. In this chapter, we focus on a more complex problem of non-bipartite matchings. We motivate this problem using an application in computational biology, disulfide connectivity prediction, but non-bipartite matchings can be used for many other tasks.

Identifying disulfide bridges formed by cysteine residues is critical in determining the structure of proteins. Recently proposed models have formulated this prediction task as a maximum weight perfect matching problem in a graph containing cysteines as nodes with edge weights measuring the attraction strength of the potential bridges. We exploit combinatorial properties of the perfect matching problem to define a compact, convex, quadratic program. We use kernels to efficiently learn very rich (in-fact, infinite-dimensional) models and present experiments on standard protein databases, showing that our framework achieves state-of-the-art performance on the task.

Throughout this chapter, we use the problem of disulfide connectivity prediction as an example. We provide some background on this problem.

10.1 Disulfide connectivity prediction

Proteins containing cysteine residues form intra-chain covalent bonds known as *disulfide bridges*. Such bonds are a very important feature of protein structure since they enhance conformational stability by reducing the number of configurational states and decreasing the entropic cost of folding a protein into its native state [Matsumura *et al.*, 1989]. They do so mostly by imposing strict structural constraints due to the resulting links between distant regions of the protein sequence [Harrison & Sternberg, 1994].

Knowledge of the exact disulfide bonding pattern in a protein provides information about protein structure and possibly its function and evolution. Furthermore, since the disulfide connectivity pattern imposes structure constraints, it can be used to reduce the search space in both protein folding prediction as well as protein 3D structure prediction. Thus, the development of efficient, scalable and accurate methods for the prediction of disulfide bonds has numerous practical applications.

Recently, there has been increased interest in applying computational techniques to the task of predicting the intra-chain disulfide connectivity [Fariselli & Casadio, 2001; Fariselli *et al.*, 2002; Vullo & Frasconi, 2004; Klepeis & Floudas, 2003; Baldi *et al.*, 2004]. Since a sequence may contain any number of cysteine residues, which may or may not participate in disulfide bonds, the task of predicting the connectivity pattern is typically decomposed into two subproblems: predicting the bonding state of each cysteine in the sequence, and predicting the exact connectivity among bonded cysteines. Alternatively, there are methods [Baldi *et al.*, 2004] that predict the connectivity pattern without knowing the bonding state of each cysteine[1].

We predict the connectivity pattern by finding the maximum weighted matching in a graph in which each vertex represents a cysteine residue, and each edge represents the "attraction strength" between the cysteines it connects [Fariselli & Casadio, 2001]. We parameterize the this attraction strength via a linear combination of features, which can include the protein sequence around the two residues, evolutionary information in the form of multiple alignment profiles, secondary structure or solvent accessibility information, etc.

[1]We thank Pierre Baldi and Jianlin Cheng for introducing us to the problem of disulfide connectivity prediction and providing us with preliminary draft of their paper and results of their model, as well as the protein datasets.

10.2 Learning to match

Formally, we seek a function $h : \mathcal{X} \mapsto \mathcal{Y}$ that maps inputs $\mathbf{x} \in \mathcal{X}$ to output matchings $\mathbf{y} \in \mathcal{Y}$, for example, $\mathcal{X}$ is the space of protein sequences and $\mathcal{Y}$ is the space of matchings of their cysteines. The space of matchings $\mathcal{Y}$ is very large, in fact, superexponential in the number of nodes in a graph. However, $\mathcal{Y}$ has interesting and complex combinatorial structure which we exploit to learn h efficiently.

The training data consists of m examples $S = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{m}$ of input graphs and output matchings. We assume that the input $\mathbf{x}$ defines the space of possible matchings using some deterministic procedure. For example, given a protein sequence, we construct a complete graph where each node corresponds to a cysteine. We represent each possible edge between nodes j and k ($j < k$) in example i using a binary variable $y_{jk}^{(i)}$. For simplicity, we assume complete graphs, but very little needs to be changed to handle sparse graphs.

If example i has L_i nodes, then there are $L_i(L_i - 1)/2$ edge variables, so $\mathbf{y}^{(i)}$ is a binary vector of dimension $L_i(L_i - 1)/2$. In a perfect matching, each node is connected *exactly* one other node. In non-perfect matchings, each node is connected to *at most* one other node. Let $n_i = L_i/2$, then for complete graphs with even number of vertices L_i, the number of possible perfect matchings is $\frac{(2n_i)!}{2^{n_i} n_i!}$ (which is $\Omega((\frac{n_i}{2})^{n_i})$, super-exponential in n_i). For example, 1ANS protein in Fig. 10.1 has 6 cysteines (nodes), 15 potential bonds (edges) and 15 possible perfect matchings.

Our hypothesis class is maximum weight matchings:

$$h_s(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} \sum_{jk} s_{jk}(\mathbf{x}) y_{jk}, \tag{10.1}$$

For disulfide connectivity prediction, this model was used by Fariselli and Casadio [2001]. Their model assigns an attraction strength $s_{jk}(\mathbf{x})$ to each pair of cysteines, calculated by assuming that all residues in the local neighborhoods of the two cysteines make contact, and summing contact potentials for pairs of residues. We consider a simple but very general class of attraction scoring functions defined by a weighted combination of features or *basis functions*:

$$s_{jk}(\mathbf{x}) = \sum_{d} w_d f_d(\mathbf{x}_{jk}) = \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}), \tag{10.2}$$

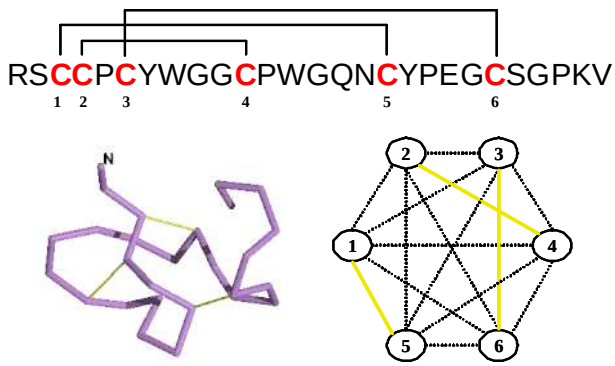


Figure 10.1: PDB protein 1ANS: amino acid sequence, 3D structure, and graph of potential disulfide bonds. Actual disulfide connectivity is shown in yellow in the 3D model and the graph of potential bonds.

where $\mathbf{x}_{jk}$ is the portion of the input $\mathbf{x}$ that directly relates to nodes j and k, $f_d(\mathbf{x}_{jk})$ is a real-valued basis function and $w_d \in \mathbb{R}$. For example, the basis functions can represent arbitrary information about the two cysteine neighborhoods: the identity of the residues at specific positions around the two cysteines, or the predicted secondary structure in the neighborhood of each cysteine. We assume that the user provides the basis functions, and that our goal is to learn the weights $\mathbf{w}$, for the model:

$$h_{\mathbf{w}}(\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y}} \sum_{jk} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}) y_{jk}. \tag{10.3}$$

Below, we will abbreviate $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}) \equiv \sum_{jk} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}) y_{jk}$, and $\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) \equiv \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y})$,

The naive formulation of the max-margin estimation, which enumerates all perfect matchings for each example i, is:

$$\min \ \frac{1}{2}||\mathbf{w}||^2 \quad \text{s.t.} \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y}), \quad \forall i, \ \forall \mathbf{y} \in \mathcal{Y}^{(i)}. \tag{10.4}$$

The number of constraints in this formulation is super-exponential in the number of nodes in each example. In the following sections we present two max-margin formulations, first with an exponential set of constraints (Sec. 10.3), and then with a polynomial one (Sec. 10.4).

10.3 Min-max formulation

Using the min-max formulation from Ch. 4, we have a single $\max$ constraint for each i:

$$\min \ \frac{1}{2}||\mathbf{w}||^2 \quad \text{s.t.} \quad \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) \geq \max_{\mathbf{y}\in\mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})], \quad \forall i. \tag{10.5}$$

The key to solving this problem efficiently is the *loss-augmented* inference $\max_{\mathbf{y}\in\mathcal{Y}^{(i)}}[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$. Under the assumption of Hamming distance loss (or any loss function that can be written as a sum of terms corresponding to edges), this maximization is equivalent (up to a constant term) to a maximum weighted matching problem. Note that since the y variables are binary, the Hamming distance between $\mathbf{y}^{(i)}$ and $\mathbf{y}$ can be written as $(\mathbf{1}-\mathbf{y})^\top \mathbf{y}^{(i)} + (\mathbf{1}-\mathbf{y}^{(i)})^\top \mathbf{y} = \mathbf{1}^\top \mathbf{y}^{(i)} + (\mathbf{1}-2\mathbf{y}^{(i)})^\top \mathbf{y}$. Hence, the maximum weight matching where edge jk has weight $\mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}^{(i)}) + (1-2y_{jk}^{(i)})$ (plus the constant $\mathbf{1}^\top \mathbf{y}^{(i)}$) gives the value of $\max_{\mathbf{y}\in\mathcal{Y}^{(i)}}[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$.

This problem can be solved in $\mathcal{O}(L^3)$ time [Gabow, 1973; Lawler, 1976]. It can also be solved as a linear program, where we introduce continuous variables $\mu_{i,jk}$ instead of binary variables $y_{jk}^{(i)}$.

$$\begin{aligned}
\max \quad & \sum_{jk} \mu_{i,jk}[\mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}^{(i)}) + (1-2y_{jk}^{(i)})] \\
\text{s.t.} \quad & \mu_{i,jk} \geq 0, \quad 1 \leq j < k \leq L_i; \qquad \sum_k \mu_{i,jk} \leq 1, \quad 1 \leq j \leq L_i; \\
& \sum_{j,k\in V} \mu_{i,jk} \leq \frac{1}{2}(|V|-1), \quad V \subseteq \{1,\ldots,L_i\},\ |V| \geq 3 \text{ and odd.}
\end{aligned} \tag{10.6}$$

The constraints $\sum_k \mu_{i,jk} \leq 1$ require that the number of bonds incident on a node is less or equal to one. For perfect matchings, these constraints are changed to $\sum_k \mu_{i,jk} = 1$ to ensure exactly one bond. The subset constraints (in the last line of Eq. (10.6)) ensure that solutions to the LP are integral [Edmonds, 1965]. Note that we have an exponential number of constraints ($\mathcal{O}(2^{(L_i-1)})$), but this number is asymptotically smaller than the number of possible matchings . It is an open problem to derive a polynomial sized LP formulation for perfect matchings [Schrijver, 2003].

We can write the loss-augmented inference problem in terms of the LP in Eq. (10.6):

$$\max_{\mathbf{y}\in\mathcal{Y}^{(i)}}[\mathbf{w}^\top\mathbf{f}_i(\mathbf{y})+\ell_i(\mathbf{y})] = d_i + \max_{\substack{\mathbf{A}_i\mu_i\leq\mathbf{b}_i\\ \mu_i\geq 0}} \mu_i^\top[\mathbf{F}_i\mathbf{w}+\mathbf{c}_i],$$

where: $d_i = \mathbf{1}^\top\mathbf{y}^{(i)}$; μ_i is a vector of length $L_i(L_i-1)/2$ indexed by bond jk; $\mathbf{A}_i$ and $\mathbf{b}_i$ are the appropriate constraint coefficient matrix and right hand side vector, respectively. $\mathbf{F}_i$ is a matrix of basis function coefficients such that the component jk of the vector $\mathbf{F}_i\mathbf{w}$ is $\mathbf{w}^\top\mathbf{f}(\mathbf{x}_{jk}^{(i)})$ and $\mathbf{c}_i = (\mathbf{1}-2\mathbf{y}^{(i)})$. Note that the dependence on $\mathbf{w}$ is linear and occurs only in the objective of the LP.

The dual of the LP in Eq. (10.6) is

$$\min \quad \lambda_i^\top\mathbf{b}_i \quad \text{s.t.} \quad \mathbf{A}_i^\top\lambda_i \geq \mathbf{F}_i\mathbf{w}+\mathbf{c}_i; \quad \lambda_i \geq 0. \tag{10.7}$$

We plug it into Eq. (10.5) and combine the minimization over λ with minimization over $\mathbf{w}$.

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{s.t.} \quad & \mathbf{w}^\top\mathbf{f}_i(\mathbf{y}^{(i)}) \geq d_i + \lambda_i^\top\mathbf{b}_i, \quad \forall i; \\ & \mathbf{A}_i^\top\lambda_i \geq \mathbf{F}_i\mathbf{w}+\mathbf{c}_i, \quad \forall i; \\ & \lambda_i \geq 0, \quad \forall i. \end{aligned} \tag{10.8}$$

In case that our basis functions are not rich enough to predict the training data perfectly, we can introduce a slack variable ξ_i for each example i to allow violations of the constraints and minimize the sum of the violations:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top\mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq d_i + \lambda_i^\top\mathbf{b}_i, \quad \forall i; \\ & \mathbf{A}_i^\top\lambda_i \geq \mathbf{F}_i\mathbf{w}+\mathbf{c}_i, \quad \forall i; \\ & \lambda_i \geq 0, \quad \forall i; \qquad \xi_i \geq 0, \quad \forall i. \end{aligned} \tag{10.9}$$

The parameter C allows the user to trade off violations of the constraints with fit to the

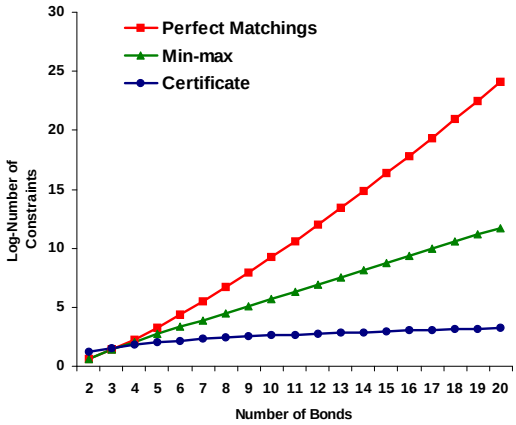


Figure 10.2: Log of the number of QP constraints (y-axis) vs. number of bonds (x-axis) in the three formulations (perfect matching enumeration, min-max and certificate).

data.

Our formulation is a linearly-constrained quadratic program, albeit with an exponential number of constraints. In the next section, we develop an equivalent polynomial size formulation.

10.4 Certificate formulation

Rather than solving the loss-augmented inference problem explicitly, we can focus on finding a compact *certificate of optimality* that guarantees that $\mathbf{y}^{(i)} = \arg\max_{\mathbf{y}}[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$. We consider perfect matchings and then provide a reduction for the non-perfect case. Let M be a perfect matching for a complete undirected graph $G = (V, E)$. In an *alternating cycle/path* in G with respect to M, the edges alternate between those that belong to M and those that do not. An alternating cycle is *augmenting* with respect to M if the score of the edges in the matching M is smaller that the score of the edges not in the matching M.

Theorem 10.4.1 ***[Edmonds, 1965]*** *A perfect matching M is a maximum weight perfect matching if and only if there are no augmenting alternating cycles.*

The number of alternating cycles is exponential in the number of vertices, so simply enumerating all of them will not do. Instead, we can rule out such cycles by considering shortest paths.

We begin by negating the score of those edges not in M. In the discussion below we assume that each edge score s_{jk} has been modified this way. We also refer to the score s_{jk} as the length of the edge jk. An alternating cycle is augmenting if and only if its length is negative. A condition ruling out negative length alternating cycles can be stated succinctly using a kind of distance function. Pick an arbitrary root node r. Let d_j^e, with $j \in V$, $e \in \{0, 1\}$, denote the length of the shortest distance alternating path from r to j, where $e = 1$ if the last edge of the path is in M, 0 otherwise. These shortest distances are well-defined if and only if there are no negative alternating cycles. The following constraints capture this distance function.

$$\begin{aligned} s_{jk} &\geq d_k^0 - d_j^1, \qquad s_{jk} \geq d_j^0 - d_k^1, \qquad \forall\, jk \notin M; \\ s_{jk} &\geq d_k^1 - d_j^0, \qquad s_{jk} \geq d_j^1 - d_k^0, \qquad \forall\, jk \in M. \end{aligned} \tag{10.10}$$

Theorem 10.4.2 *There exists a distance function $\{d_j^e\}$ satisfying the constraints in Eq. (10.10) if and only if no augmenting alternating cycles exist.*

Proof.

(If) Suppose there are no augmenting alternating cycles. Since any alternating paths from r to j can be shortened (or left the same length) by removing the cycles they contain, the two shortest paths to j (one ending with M-edge and one not) contain no cycles. Then let d_j^0 and d_j^1 be the length of those paths, for all j (for $j = r$, set $d_r^0 = d_r^1 = 0$). Then for any jk (or kj) in M, the shortest path to j ending with an edge not in M plus the edge jk (or kj) is an alternating path to k ending with an edge in M. This path is longer or same length as the shortest path to k ending with an edge in M: $s_{jk} + d_j^0 \geq d_k^1$ (or $s_{kj} + d_j^0 \geq d_k^1$), so the constraint is satisfied. Similarly for $jk, kj \notin M$.

(Only if) Suppose a distance function $\{d_j^e\}$ satisfies the constraints in Eq. (10.10). Consider an alternating cycle C. We renumber the nodes such that the cycle passes through nodes $1, 2, \ldots, l$ and the first edge, $(1, 2)$, is in M. The length of the path is $s(C) = s_{1,l} + \sum_{j=1}^{l-1} s_{j,j+1}$. For each odd j, the edge $(j, j+1)$ is in M, so $s_{j,j+1} \geq d_{j+1}^1 - d_j^0$. For

even j, the edge $(j, j+1)$ is not in M, so $s_{j,j+1} \geq d^0_{j+1} - d^1_j$. Finally, the last edge, $(1, l)$, is not in M, so $s_{1,l} \geq d^0_1 - d^1_l$. Summing the edges, we have:

$$s(C) \geq d^0_1 - d^1_l + \sum_{j=1,odd}^{l-1} [d^1_{j+1} - d^0_j] + \sum_{j=2,even}^{l-1} [d^0_{j+1} - d^1_j] = 0.$$

Hence all alternating cycles have nonnegative length. ∎

In our learning formulation we have the loss-augmented edge weights $s^{(i)}_{jk} = (2y^{(i)}_{jk} - 1)(\mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}) + 1 - 2y^{(i)}_{jk})$. Let $\mathbf{d}_i$ be a vector of distance variables d^e_j, $\mathbf{H}_i$ and $\mathbf{G}_i$ be matrices of coefficients and $\mathbf{q}_i$ be a vector such that $\mathbf{H}_i\mathbf{w} + \mathbf{G}_i\mathbf{d}_i \geq \mathbf{q}_i$ represents the constraints in Eq. (10.10) for example i. Then the following joint convex program in $\mathbf{w}$ and $\mathbf{d}$ computes the max-margin parameters:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{s.t.} \quad & \mathbf{H}_i\mathbf{w} + \mathbf{G}_i\mathbf{d}_i \geq \mathbf{q}_i, \quad \forall i. \end{aligned} \tag{10.11}$$

Once again, in case that our basis functions are not rich enough to predict the training data perfectly, we can introduce a slack variable vector ξ_i to allow violations of the constraints.

The case of non-perfect matchings can be handled by a reduction to perfect matchings as follows [Schrijver, 2003]. We create a new graph by making a copy of the nodes and the edges and adding edges between each node and the corresponding node in the copy. We extend the matching by replicating its edges in the copy and for each unmatched node, introduce an edge to its copy. We define $\mathbf{f}(\mathbf{x}_{jk}) \equiv 0$ for edges between the original and the copy. Perfect matchings in this graph projected onto the original graph correspond to non-perfect matchings in the original graph.

The comparison between the log-number of constraints for our three equivalent QP formulations (enumeration of all perfect matchings, min-max and certificate) is shown in Fig. 10.2. The x-axis is the number of edges in the matching (number of nodes divided by two).

10.5 Kernels

Instead of directly optimizing the primal problem in Eq. (10.8), we can work with its dual. Each training example i has $L_i(L_i - 1)/2$ dual variables, and $\alpha_{jk}^{(i)}$ is the dual variable associated with the features corresponding to the edge jk. Let $\alpha^{(i)}$ be the vector of dual variables for example i. The dual quadratic optimization problem has the form:

$$\begin{aligned} \max \quad & \sum_i \mathbf{c}_i^\top \alpha^{(i)} - \frac{1}{2} \left\| \sum_i \sum_{jk \in E^{(i)}} \left[\left(C y_{jk}^{(i)} - \alpha_{jk}^{(i)} \right) \mathbf{f}(\mathbf{x}_{jk}^{(i)}) \right] \right\|^2 \\ \text{s.t.} \quad & \mathbf{A}_i \alpha^{(i)} \leq C \mathbf{b}_i, \quad \forall i. \\ & \alpha^{(i)} \geq 0, \quad \forall i. \end{aligned} \tag{10.12}$$

The only occurrence of feature vectors is in the expansion of the squared-norm term in the objective:

$$\sum_{i,j} \sum_{kl \in E^{(i)}} \sum_{mn \in E^{(j)}} \left(C y_{jk}^{(i)} - \alpha_{jk}^{(i)} \right) \mathbf{f}(\mathbf{x}_{kl}^{(i)})^\top \mathbf{f}(\mathbf{x}_{mn}^{(j)}) \left(C y_{jk}^{(j)} - \alpha_{jk}^{(j)} \right) \tag{10.13}$$

Therefore, we can apply the kernel trick and let $\mathbf{f}(\mathbf{x}_{kl}^{(i)})^\top \mathbf{f}(\mathbf{x}_{mn}^{(j)}) = K(\mathbf{x}_{kl}^{(i)}, \mathbf{x}_{mn}^{(j)})$. Thus, we can efficiently map the original features $\mathbf{f}(\mathbf{x}_{jk})$ to a high-dimensional space. The primal and dual solutions are related by:

$$\mathbf{w} = \sum_i \sum_{jk} (C y_{jk}^{(i)} - \alpha_{jk}^{(i)}) \mathbf{f}(\mathbf{x}_{jk}^{(i)}) \tag{10.14}$$

Eq. (10.14) can be used to compute the attraction strength $s_{jk}(\mathbf{x})$ in a kernelized manner at prediction time. The polynomial-sized representation in Eq. (10.11) is similarly kernelizable.

10.6 Experiments

We assess the performance of our method on two datasets containing sequences with experimentally verified bonding patterns: DIPRO2 and SP39. The DIPRO2 dataset[2] was compiled and made publicly available by Baldi *et al.* [2004]. It consists of all proteins from PDB [Berman *et al.*, 2000], as of May 2004, which contain intra-chain disulfide bonds. After redundance reduction there are a total of 1018 sequences. In addition, the sequences are annotated with secondary structure and solvent accessibility information derived from the DSSP database [Kabsch & Sander, 1983]. The SP39 dataset is extracted from the Swiss-Prot database of proteins [Bairoch & Apweiler, 2000], release 39. It contains only sequences with experimentally verified disulfide bridges, and has a total of 726 proteins. The same dataset was used in earlier work [Baldi *et al.*, 2004; Vullo & Frasconi, 2004; Fariselli & Casadio, 2001], and we have followed the same procedure for extracting sequences from the database.

Even though our method is applicable to both sequences with a high number of bonds or sequences in which the bonding state of cysteine residues is unknown, we report results for the case where the bonding state is known, and the number of bonds is between 2 and 5 (since the case of 1 bond is trivial). The DIPRO2 contains 567 such sequences, and only 53 sequences with a higher number of bonds, so we are able to perform learning on over 90% of all proteins. There are 430 proteins with 2 and 3 bonds and 137 with 4 and 5 bonds. SP39 contains 446 sequences containing between 2 and 5 bonds.

In order to avoid biases during testing, we adopt the same dataset splitting procedure as the one used in previous work [Fariselli & Casadio, 2001; Vullo & Frasconi, 2004; Baldi *et al.*, 2004]. We split SP39 into 4 different subsets, with the constraint that proteins no proteins with sequence similarity of more than 30% belong to different subsets. Sequence similarity was derived using an all-against-all rigorous Smith-Waterman local pairwise alignment [Smith & Waterman, 1981] (with the BLOSUM65 scoring matrix, gap penalty 12 and gap extension 4). Pairs of chains whose alignment is less than 30 residues were considered unrelated. The DIPRO2 dataset was split similarly into 5 folds, although the procedure had less effect due to the redundance reduction applied by the authors of the

[2] http://contact.ics.uci.edu/bridge.html

dataset.

Models

The experimental results we report use the dual formulation of Sec. 10.5 and an RBF kernel $K(\mathbf{x}_{jk}, \mathbf{x}_{lm}) = \exp(\frac{\|\mathbf{x}_{jk}-\mathbf{x}_{lm}\|^2}{\gamma})$, with $\gamma \in [0.1, 10]$. We use the exponential sized representation of Sec. 10.3 since for the case of proteins containing between two and five bonds, it is more efficient due to the low constants in the exponential problem size. We used commercial QP software (CPLEX) to train our models. Training time took around 70 minutes for 450 examples, using a sequential optimization procedure which solves QP subproblems associated with blocks of training examples. We are currently working on an implementation of the certificate formulation Sec. 10.4 to handle longer sequences and non-perfect matchings (when bonding state is unknown). Below, we describe several models we used.

The features we experimented with were all based on the local regions around candidate cysteine pairs. For each pair of candidate cysteines $\{j, k\}$, where $j < k$, we extract the amino-acid sequence in windows of size n centered at j and k. As in Baldi *et al.* [2004], we augment the features of each model with the number of residues between j and k. The models below use windows of size $n = 9$.

The first model, *SEQUENCE*, uses the features described above: for each window, the actual sequence is expanded to a $20 \times n$ binary vector, in which the entries denote whether or not a particular amino acid occurs at the particular position. For example, the 21st entry in the vector represents whether or not the amino-acid Alanine occurs at position 2 of the local window, counting from the left end of the window. The final set of features for each $\{j, k\}$ pair of cysteines is simply the two local windows concatenated together, augmented with the linear distance between the cysteine residues.

The second model, *PROFILE*, is the same as *SEQUENCE*, except that instead of using the actual protein sequence, we use multiple sequence alignment profile information. Multiple alignments were computed by running PSI-BLAST using default settings to align the sequence with all sequences in the NR database [Altschul *et al.*, 1997]. Thus, the input at each position of a local window is the frequency of occurrence of each of the 20 amino-acids in the alignments.

K	*PROFILE*	*DAG-RNN*
2	**0.75** / **0.75**	0.74 / 0.74
3	0.60 / 0.48	**0.61** / **0.51**
4	**0.46** / 0.24	0.44 / **0.27**
5	**0.43** / **0.16**	0.41 / 0.11

(a)

K	*SEQUENCE*	*PROFILE*	*PROFILE-SS*
2	0.70 / 0.70	0.73 / 0.73	0.79 / 0.79
3	0.62 / 0.52	0.67 / 0.59	0.74 / 0.69
4	0.44 / 0.21	0.59 / 0.44	0.70 / 0.56
5	0.29 / 0.06	0.43 / 0.17	0.62 / 0.27

(b)

Table 10.1: Numbers indicate *Precision / Accuracy*. (a) Performance of *PROFILE* model on SP39 vs. preliminary results of the DAG-RNN model [Baldi *et al.*, 2004] which represent the best currently published results. In each row, the best performance is in **bold**. (b) Performance of *SEQUENCE*, *PROFILE*, *PROFILE-SS* models on the DIPRO2 dataset.

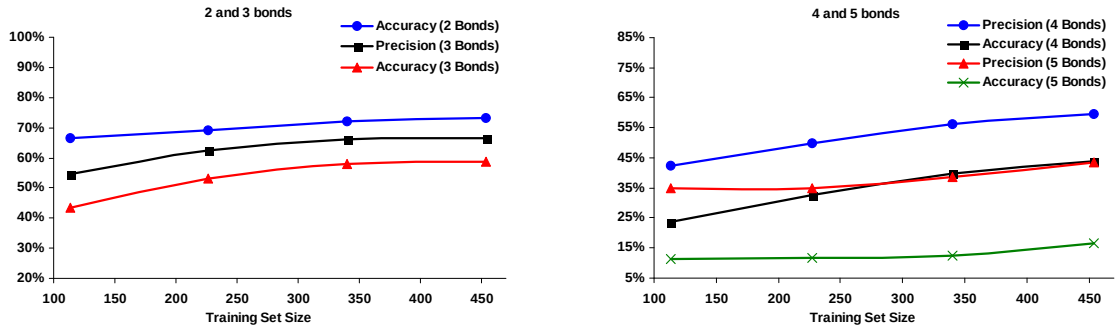
The third model, *PROFILE-SS*, augments the *PROFILE* model with secondary structure and solvent-accessibility information. The DSSP program produces 8 types of secondary structure, so we augment each local window of size n with an additional length $8 \times n$ binary vector, as well as a length n binary vector representing the solvent accessibility at each position.

Results and discussion

We evaluate our algorithm using two metrics: accuracy and precision. The accuracy measure counts how many full connectivity patterns were predicted correctly, whereas precision measures the number of correctly predicted bonds as a fraction of the total number of possible bonds.

The first set of experiments compares our model to preliminary results reported in Baldi *et al.* [2004], which represent the current top-performing system. We perform 4-fold cross-validation on SP39 in order to replicate their setup. As Table 10.1 shows, the *PROFILE* model achieves comparable results, with similar or better levels of precision for all bond numbers, and slightly lower accuracies for the case of 2 and 3 bonds.

In another experiment, we show the performance gained by using multiple alignment information by comparing the results of the *SEQUENCE* model with the *PROFILE*. As we can see from Table 10.1(b), the evolutionary information captured by the amino-acid alignment frequencies plays an important role in increasing the performance of the algorithm.



(a) (b)

Figure 10.3: Performance of *PROFILE* model as training set size changes for proteins with (a) 2 and 3 bonds (b) 4 and 5 bonds.

The same phenomenon is observed by Vullo and Frasconi [2004] in their comparison of sequence and profile-based models.

As a final experiment, we examine the role that secondary structure and solvent-accessibility information plays in the model *PROFILE-SS*. Table 10.1(b) shows that the gains are significant, especially for sequences with 3 and 4 bonds. This highlights the importance of developing even richer features, perhaps through more complex kernels.

Fig. 10.3 shows the performance of the *PROFILE* model as training set size grows. We can see that for sequences of all bond numbers, both accuracy and precision increase as the amount of data grows. The trend is more pronounced for sequences with 4 and 5 bonds because they are sparsely distributed in the dataset. Such behavior is very promising, since it validates the applicability of our algorithm as the availability of high-quality disulfide bridge annotations increases with time.

10.7 Related work

The problem of inverse perfect matching has been studied by Liu and Zhang [2003] in the inverse combinatorial optimization framework we describe in Sec. 4.3: Given a set of nominal weights $\mathbf{w}^0$ and a perfect matching M, which is not a maximum one with respect to $\mathbf{w}^0$, find a new weight vector $\mathbf{w}$ that makes M optimal and minimizes $||\mathbf{w}^0 - \mathbf{w}||_p$ for $p = 1, \infty$. They do not provide a compact optimization problem for this related but

different task, relying instead on the ellipsoid method with constraint generation.

The problem of disulfide bond prediction first received comprehensive computational treatment in Fariselli and Casadio [2001]. They modeled the prediction problem as finding a perfect matching in a weighted graph where vertices represent bonded cysteine residues, and edge weights correspond to attraction strength. The problem of learning the edge weights was addressed using a simulated annealing procedure. Their method is only applicable to the case when bonding state is known. In Fariselli *et al.* [2002], the authors switch to using a neural network for learning edge weights and achieve better performance, especially for the case of 2 and 3 disulfide bonds.

The method in Vullo and Frasconi [2004] takes a different approach to the problem. It scores candidate connectivity patterns according to their similarity with respect to the correct pattern, and uses a recursive neural network architecture [Frasconi *et al.*, 1998] to score candidate patterns. At prediction time the pattern scores are used to perform an exhaustive search on the space of all matchings. The method is computationally limited to sequences of 2 to 5 bonds. It also uses multiple alignment profile information and demonstrates its benefits over sequence information.

In Baldi *et al.* [2004], the authors achieve the current state-of-the-art performance on the task. Their method uses Directed Acyclic Graph Recursive Neural Networks [Baldi & Pollastri, 2003] to predict bonding probabilities between cysteine pairs. The prediction problem is solved using a weighted graph matching based on these probabilities. Their method performs better than the one in Vullo and Frasconi [2004] and is also the only one which can cope with sequences with more than 5 bonds. It also improves on previous methods by not assuming knowledge of bonding state.

A different approach to predicting disulfide bridges is reported in Klepeis and Floudas [2003], where bond prediction occurs as part of predicting β-sheet topology in proteins. Residue-to-residue contacts (which include disulphide bridges) are predicted by solving a series of constrained integer programming problems. Interestingly, the approach can be used to predict disulfide bonds with no knowledge of bonding state, but the results are not comparable with those in other publications.

The task of predicting whether or not a cysteine is bonded has also been addressed using a variety of machine learning techniques including neural networks, SVMs, and HMMs

[Fariselli *et al.*, 1999; Fiser & Simon, 2000; Martelli *et al.*, 2002; Frasconi *et al.*, 2002; Ceroni *et al.*, 2003] Currently the top performing systems have accuracies around 85%.

10.8 Conclusion

In this chapter, we derive a compact convex quadratic program for the problem of learning to match. Our approach learns a parameterized scoring function that reproduces the observed matchings in a training set. We present two formulations: one which is based on a linear programming approach to matching, requiring an exponential number of constraints, and one which develops a certificate of matching optimality for a compact polynomial-sized representation. We apply our framework to the task of disulfide connectivity prediction, formulated as a weighted matching problem. Our experimental results show that the method can achieve performance comparable to current top-performing systems. Furthermore, the use of kernels makes it easy to incorporate rich sets of features such as secondary structure information, or extended local neighborhoods of the protein sequence. In the future, it will be worthwhile to examine how other kernels, such as convolution kernels for protein sequences, will affect performance. We also hope to explore the more challenging problem of disulfide connectivity prediction when the bonding state of cysteines is unknown. While we have developed the framework to handle that task, it remains to experimentally determine how well the method performs, especially in comparison to existing methods [Baldi *et al.*, 2004], which have already addressed the more challenging setting.

Chapter 11

Correlation clustering

Data can often be grouped in many different reasonable clusterings. For example, one user may organize her email messages by project and time, another by sender and topic. Images can be segmented by hue or object boundaries. For a given application, there might be only one of these clusterings that is desirable. Learning to cluster considers the problem of finding desirable clusterings on new data, given example desirable clusterings on training data.

We focus on correlation clustering, a novel clustering method that has recently enjoyed significant attention from the theoretical computer science community [Bansal *et al.*, 2002; Demaine & Immorlica, 2003; Emanuel & Fiat, 2003]. It is formulated as a vertex partitioning problem: Given a graph with real-valued edge scores (both positive and negative), partition the vertices into clusters to maximize the score of intra-cluster edges, and minimize the weight of inter-cluster edges. Positive edge weights represent correlations between vertices, encouraging those vertices to belong to a common cluster; negative weights encourage the vertices to belong to different clusters. Unlike most clustering formulations, correlation clustering does not require the user to specify the number of clusters nor a distance threshold for clustering; both of these parameters are effectively chosen to be the best possible by the problem definition. These properties make correlation clustering a promising approach to many clustering problems; in machine learning, it has been successfully applied to coreference resolution for proper nouns [McCallum & Wellner, 2003].

Recently, several algorithms based on linear programming and positive-semidefinite

programming relaxations have been proposed to approximately solve this problem. In this chapter, we employ these relaxations to derive a max-margin formulation for learning the edge scores for correlation clustering from clustered training data. We formulate the approximate learning problem as a compact convex program with quadratic objective and linear or positive-semidefinite constraints. Experiments on synthetic and real-world data show the ability of the algorithm to learn an appropriate clustering metric for a variety of desired clusterings.

11.1 Clustering formulation

An instance of correlation clustering is specified by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes and edge score s_{jk} for each jk in $\mathcal{E}$, ($j < k$). We assume that the graph is fully connected (if it is not, we can make it fully connected by adding appropriate edges jk with $s_{jk} = 0$). We define binary variables y_{jk}, one for each edge jk, that represent whether node j and k belong to the same cluster. Let $\mathcal{Y}$ be the space of assignments $\mathbf{y}$ that define legal partitions. For notational convenience, we introduce both y_{jk} and y_{kj} variables, which will be constrained to have the same value. We also introduce y_{jj} variables, and fix them to have value 1 and set $s_{jj} = 0$.

Bansal *et al.* [2002] consider two related problems:

$$\max_{\mathbf{y}\in\mathcal{Y}} \sum_{jk:s_{jk}>0} s_{jk}y_{jk} - \sum_{jk:s_{jk}<0} s_{jk}(1-y_{jk}); \qquad \text{(MAXAGREE)}$$

$$\min_{\mathbf{y}\in\mathcal{Y}} \sum_{jk:s_{jk}>0} s_{jk}(1-y_{jk}) - \sum_{jk:s_{jk}<0} s_{jk}y_{jk}; \qquad \text{(MINDISAGREE)}$$

The motivation for the names of the two problems comes from separating the set of edges into positive weight edges and negative weight edges. The best score is obviously achieved by including all the positive and excluding all the negative edges, but this will not generally produce a valid partition. In MAXAGREE, we maximize the "agreement" of the partition with the positive/negative designations: the weight of the *positive included* edges minus the weight of *negative excluded* edges. In MINDISAGREE, we minimize the disagreement: the weight of *positive excluded* edges minus the weight of *negative included* edges. In

particular, let

$$s^* = \max_{\mathbf{y}\in\mathcal{Y}} \sum s_{jk} y_{jk}; \qquad s^- = \sum_{jk:s_{jk}<0} s_{jk}; \qquad s^+ = \sum_{jk:s_{jk}>0} s_{jk}.$$

Then the value of MAXAGREE is $s^* - s^-$ and the value of MINDISAGREE $s^+ - s^*$. The optimal partition for the two problems is of course the same (if it is unique). Bansal *et al.* [2002] show that both of these problems are NP-hard (but have different approximation hardness). We will concentrate on the maximization version, MAXAGREE. Several approximation algorithms have been developed based on Linear and Semidefinite Programming [Charikar *et al.*, 2003; Demaine & Immorlica, 2003; Emanuel & Fiat, 2003], which we consider in the next sections.

11.1.1 Linear programming relaxation

In order to insure that $\mathbf{y}$ defines a partition, it is sufficient to enforce a kind of triangle inequality for each triple of nodes $j < k < l$:

$$y_{jk} + y_{kl} \leq y_{jl} + 1; \qquad y_{jk} + y_{jl} \leq y_{kl} + 1; \qquad y_{jl} + y_{kl} \leq y_{jk} + 1. \tag{11.1}$$

The triangle inequality enforces transitivity: if j and k are in the same cluster ($y_{jk} = 1$) and k and l are in the same cluster ($y_{kl} = 1$), then j and l will be forced to be in this cluster ($y_{jl} = 1$). The other two cases are similar. Any symmetric, transitive binary relation induces a partition of the objects.

With these definitions, we can express the MAXAGREE problem as an integer linear program (ignoring the constant $-s^-$):

$$\begin{aligned} \max \quad & \sum_{jk} s_{jk} y_{jk} \\ \text{s.t.} \quad & y_{jk} + y_{kl} \leq y_{lj} + 1, \quad \forall j,k,l; \qquad y_{jj} = 1, \quad \forall j; \qquad y_{jk} \in \{0,1\}, \quad \forall j,k. \end{aligned} \tag{11.2}$$

Note that the constraints imply that $y_{ab} = y_{ba}$ for any two nodes a and b. To see this, consider the inequalities involving node a and b with $j = a, k = b, l = b$ and $j = b, k =$

$a, l = a$:

$$y_{ab} + y_{bb} \leq y_{ba} + 1; \qquad y_{ba} + y_{aa} \leq y_{ab} + 1;$$

Since $y_{aa} = y_{bb} = 1$, we have $y_{ab} = y_{ba}$.

The LP relaxation is obtained by replacing the binary variables $y_{jk} \in \{0, 1\}$ in Eq. (11.2) with continuous variables $0 \leq \mu_{jk} \leq 1$.

$$\mu_{jk} + \mu_{kl} \leq \mu_{lj} + 1, \quad \forall j, k, l; \qquad \mu_{jj} = 1, \quad \forall j; \qquad \mu_{jk} \geq 0, \quad \forall j, k. \tag{11.3}$$

Note that $\mu_{jk} \leq 1$ is implied since the triangle inequality with $j = l$ gives $\mu_{jk} + \mu_{kj} \leq \mu_{jj} + 1$, and since $\mu_{jj} = 1$ and $\mu_{jk} = \mu_{kj}$, we have $\mu_{jk} \leq 1$.

We are not guaranteed that this relaxation will produce integral solutions. The LP solution, s^{LP}, is an upper bound on the s^*. Charikar *et al.* [2003] show that the integrality gap of this upper bound is at least $2/3$:

$$\frac{s^* - s^-}{s^{LP} - s^-} < \frac{2}{3}.$$

11.1.2 Semidefinite programming relaxation

An alternative formulation [Charikar *et al.*, 2003] is the SDP relaxation. Let $\mathrm{mat}(\mu)$ denote the variables μ_{jk} arranged into a matrix.

$$\begin{aligned} \max \quad & \sum_{jk} s_{jk}\mu_{jk} \\ \text{s.t.} \quad & \mathrm{mat}(\mu) \succeq 0; \qquad \mu_{jj} = 1, \quad \forall j; \qquad \mu_{jk} \geq 0, \quad \forall j, k. \end{aligned} \tag{11.4}$$

In effect, we substituted the triangle inequalities by the semidefinite constraint. To motivate this relaxation, consider any clustering solution. Choose a collection of orthogonal unit vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_K\}$, one for each cluster in the solution. Every vertex j in the cluster is assigned the unit vector $\mathbf{v}_j$ corresponding to the cluster it is in. If vertices j and k are in the same cluster, then $\mathbf{v}_j^\top \mathbf{v}_k = 1$, if not, $\mathbf{v}_j^\top \mathbf{v}_k = 0$. The score of the clustering

solution can now be expressed in terms of the dot products $\mathbf{v}_j^\top \mathbf{v}_k$. In the SDP relaxation in Eq. (11.4), we have $\mathrm{mat}(\mu) \succeq 0$, which can be decomposed into a sum of outer products $\mathrm{mat}(\mu) = \sum_j \mathbf{v}_j \mathbf{v}_j^\top$.

The entries μ_{jk} correspond to inner products $\mathbf{v}_j^\top \mathbf{v}_k$. The vectors generating the inner products are unit vectors by the requirement $\mu_{jj} = 1$. However, they do not necessarily form a set of orthogonal vectors.

The SDP solution, s^{SDP}, is also an upper bound on the s^* and Charikar *et al.* [2003] show that the integrality gap of this upper bound is at least 0.82843:

$$\frac{s^* - s^-}{s^{SDP} - s^-} \leq 0.82843.$$

11.2 Learning formulation

The score for an edge is commonly derived from the characteristics of the pair of nodes. Specifically, we parameterize the score as a weighted combination of basis functions

$$s_{jk} = \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}),$$

$\mathbf{w}, \mathbf{f}(\mathbf{x}_{jk}) \in \mathbb{R}^n$, where $\mathbf{x}_{jk}$ is a set of features associated with nodes j and k. In document clustering, the entries of $\mathbf{f}_{\mathbf{x}_{jk}}$ might be the words shared by the documents j and k, while if one is clustering points in $\mathbb{R}^n$, features might be distances along different dimensions. We assume $\mathbf{f}(\mathbf{x}_{jj}) = 0$ so that $s_{jj} = 0$. Hence we write

$$\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{jk} y_{jk} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}).$$

Furthermore, we assume that the loss function decomposes over the edges, into a sum of edge losses $\ell_{i,jk}(y_{jk})$:

$$\ell_i(\mathbf{y}) = \sum_{jk} \ell_{i,jk}(y_{jk}) = \sum_{jk} y_{jk} \ell_{i,jk}(1) + (1 - y_{jk}) \ell_{i,jk}(0) = \ell_i(0) + \sum_{jk} y_{jk} \ell_{i,jk},$$

where $\ell_{i,jk} = \ell_{i,jk}(1) - \ell_{i,jk}(0)$. For example, the Hamming loss counts the number of

edges incorrectly cut or uncut by a partition.

$$\ell_i^H(\mathbf{y}) = \sum_{jk} \mathbb{1}(y_{jk} \neq y_{jk}^{(i)}) = \sum_{jk} y_{jk}^{(i)} + \sum_{jk} y_{jk}(1 - 2y_{jk}^{(i)}) = \ell_i^H(0) + \sum_{jk} y_{jk}\ell_{i,jk},$$

where $\ell_{i,jk} = 1 - 2y_{jk}^{(i)}$.

With this assumption, the loss augmented maximization is

$$\ell_i(0) + \max_{\mathbf{y}\in\mathcal{Y}} \sum_{jk} y_{jk}[\mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}^{(i)}) + \ell_{i,jk}]. \tag{11.5}$$

We can now use the LP relaxation in Eq. (11.3) and the SDP relaxation in Eq. (11.4) as upper bounds on Eq. (11.5). We use these upper-bounds in the min-max formulation to achieve approximate max-margin estimation.

The dual of the LP based upper bound for example i is $\ell_i(0)+$

$$\begin{aligned}
\min \quad & \sum_{jkl} \lambda_{i,jkl} + \sum_j z_{i,j} \\
\text{s.t.} \quad & \sum_l [\lambda_{i,jkl} + \lambda_{i,ljk} - \lambda_{i,klj}] \geq \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}^{(i)}) + \ell_{i,jk}, \quad \forall j \neq k; \\
& z_{i,j} + \sum_l [\lambda_{i,jjl} + \lambda_{i,ljj} - \lambda_{i,jlj}] \geq \ell_{i,jj}, \quad \forall j; \\
& \lambda_{i,jkl} \geq 0, \quad \forall j, k, l.
\end{aligned} \tag{11.6}$$

Above, we introduced a dual variable $\lambda_{i,jkl}$ for each triangle inequality and $z_{i,j}$ for the identity on the diagonal. Note the righthand-side of the second set of inequalities follows from the assumption $\mathbf{f}(\mathbf{x}_{jj}) = 0$.

Plugging this dual into the min-max formulation, we have:

$$
\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum \xi_i \\
\text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) + \xi_i \geq \ell_i(0) + \sum_{jkl} \lambda_{i,jkl} + \sum_j z_{i,j}, \quad \forall i; \\
& \sum_l [\lambda_{i,jkl} + \lambda_{i,ljk} - \lambda_{i,klj}] \geq \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}^{(i)}) + \ell_{i,jk}, \quad \forall i, \forall j \neq k; \\
& z_{i,j} + \sum_l [\lambda_{i,jjl} + \lambda_{i,ljj} - \lambda_{i,jlj}] \geq \ell_{i,jj}, \quad \forall i, \forall j; \\
& \lambda_{i,jkl} \geq 0, \quad \forall i, \forall j, k, l.
\end{aligned}
\tag{11.7}
$$

Similarly, the dual of the SDP based upper bound is $\ell_i(0)+$

$$
\begin{aligned}
\min \quad & \sum_j z_{i,j} \\
\text{s.t.} \quad & -\lambda_{i,jk} \geq \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}^{(i)}) + \ell_{i,jk}, \quad \forall j \neq k; \\
& z_{i,j} - \lambda_{i,jj} \geq \ell_{i,jj} \quad \forall j; \\
& \mathrm{mat}(\lambda_i) \succeq 0.
\end{aligned}
\tag{11.8}
$$

Plugging the SDP dual into the min-max formulation, we have:

$$
\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum \xi_i \\
\text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) + \xi_i \geq \ell_i(0) + \sum_j z_{i,j}, \quad \forall i; \\
& -\lambda_{i,jk} \geq \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk}^{(i)}) + \ell_{i,jk}, \quad \forall i, \forall j \neq k; \\
& z_{i,j} - \lambda_{i,jj} \geq \ell_{i,jj}, \quad \forall i, \forall j; \\
& \mathrm{mat}(\lambda_i) \succeq 0, \quad \forall i, \forall j, k, l.
\end{aligned}
\tag{11.9}
$$

11.3 Dual formulation and kernels

The dual of Eq. (11.7) and Eq. (11.9) provide some insight into the structure of the problem and enable efficient use of kernels. Here we give the dual of Eq. (11.7):

$$\begin{array}{ll} \max & \sum_{i,jk} \mu_{i,jk}\ell_{i,jk} - \frac{1}{2}C\left|\left|\sum_{i,jk}(y_{jk}^{(i)} - \mu_{i,jk})\mathbf{f}(\mathbf{x}_{jk}^{(i)})\right|\right|^2 \\ \text{s.t.} & \mu_{jk} + \mu_{kl} \leq \mu_{lj} + 1, \quad \forall i, \forall j, k, l; \quad \mu_{i,jj} = 1, \quad \forall i, \forall j; \quad \mu_{i,jk} \geq 0, \quad \forall i, \forall j, k. \end{array}$$

The dual of Eq. (11.9) is very similar, except that the linear transitivity constraints $\mu_{jk} + \mu_{kl} \leq \mu_{lj} + 1, \quad \forall i, \forall j, k, l$ are replaced by the corresponding $\mathrm{mat}(\mu_i) \succeq 0$:

$$\begin{array}{ll} \max & \sum_{i,jk} \mu_{i,jk}\ell_{i,jk} - \frac{1}{2}C\left|\left|\sum_{i,jk}(y_{jk}^{(i)} - \mu_{i,jk})\mathbf{f}(\mathbf{x}_{jk}^{(i)})\right|\right|^2 \\ \text{s.t.} & \mathrm{mat}(\mu_i) \succeq 0, \quad \forall i; \quad \mu_{i,jj} = 1, \quad \forall i, \forall j; \quad \mu_{i,jk} \geq 0, \quad \forall i, \forall j, k. \end{array}$$

The relation between the primal and dual solution is

$$\mathbf{w} = C\sum_{i,jk}(y_{jk}^{(i)} - \mu_{jk})\mathbf{f}(\mathbf{x}_{jk}^{(i)}). \tag{11.10}$$

One important consequence of this relationship is that the edge parameters are all support vector expansions. The dual objective can be expressed in terms of dot-products $\mathbf{f}(\mathbf{x}_{jk})^\top\mathbf{f}(\mathbf{x}_{lm})$. Therefore, we can use kernels $K(\mathbf{x}_{jk}, \mathbf{x}_{lm})$ to define the space of basis functions. This kernel looks at two pairs of nodes, (j, k) and (l, m), and measures the similarity between the relation between the nodes of each pair. If we are clustering points in Euclidian space, the kernel could be a function of the two segments corresponding to the pairs of points, for example, a polynomial kernel over their lengths and angle between them.

11.4 Experiments

We present experiments on a synthetic problem exploring the effect of irrelevant basis functions (features), and two real data sets, email clustering and image segmentation.

11.4.1 Irrelevant features

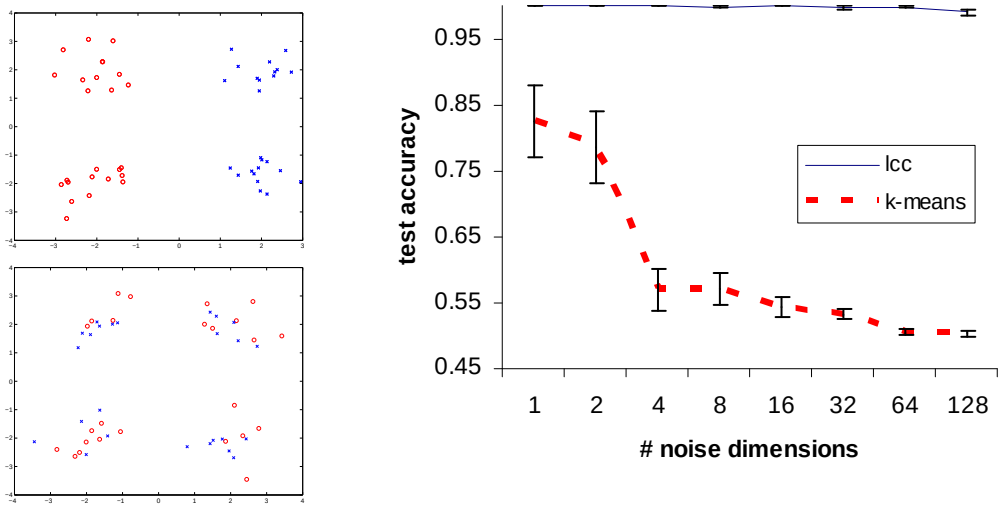
In this section, we explore on a synthetic example how our algorithm deals with irrelevant features. In particular, we generate data (100 points) from a mixture of two one-dimensional Gaussians, where each mixture component corresponds to a cluster. This first dimension is thus the relevant feature. Then we add noise components in D additional (irrelevant) dimensions. The noise is independently generated for each dimension, from a mixture of Gaussians with same difference in means and variance as for the relevant dimension. Figure 11.1(a) shows the projection of a data sample onto the first two dimensions and on two irrelevant dimensions.

Let $\mathbf{x}_j$ denote each point and $\mathbf{x}_j[d]$ denote the d-th dimension of the point. We used a basis function for each dimension $f_d(\mathbf{x}_{jk}) = e^{-(\mathbf{x}_j[d]-\mathbf{x}_k[d])^2}$, plus an additional constant basis function. The training and test data consists of a 100 samples from the model. The results in Fig. 11.1(b) illustrate the capability of our algorithm to learn to ignore irrelevant dimensions. The accuracy is the fraction of edges correctly predicted to be between/within cluster. Random clustering will give an accuracy of 50%. The comparison with k-means is simply a baseline to illustrate the effect of the noise on the data.

11.4.2 Email clustering

We also test our approach on the task of clustering email into folders. We gathered the data from the SRI CALO project.[1] Our dataset consisted of email from seven users (approximately 100 consecutive messages per user), which the users had manually filed into different folders. The number of folders for each users varied from two to six, with an average of 3-4 folders. We are interested in the problem of learning to cluster emails.

[1] `http://www.ai.sri.com/project/CALO`



(a) (b)

Figure 11.1: (a) Projection onto first two dimensions (top) and two noise dimensions (bottom); (c) Performance on 2 cluster problem as function of the number of irrelevant noise dimensions. Learning to cluster is the solid line, k-means the dashed line. Error-bars denote one standard deviation, averages are over 20 runs. Accuracy is the fraction of edges correctly predicted to be between/within cluster.

Specifically, what score function s_{jk} causes correlation clustering to give clusters similar to those that human users had chosen?

To test our learning algorithm, we use each user as a training set in turn, learning the parameters from the partition of a single user's mailbox into folders. We then use the learned parameters to cluster the other users' mail. The basis functions $\mathbf{f}(\mathbf{x}_{jk})$ measured the similarity between the text of the messages, the similarity between the "From:" field, "To:" field, and "Cc:" field. One feature was used for each common word in the pair of emails (except words that appeared in more than half the messages, which were deemed "stop words" and omitted). Also, additional features captured the proportion of shared tokens for each email field, including the from, to, Cc, subject and body fields. The algorithm is therefore able to automatically learn the relative importance of certain email fields to filing two messages together, as well as importance of meaningful words versus common, irrelevant ones.

We compare our method to the k-means clustering algorithm using with the same word

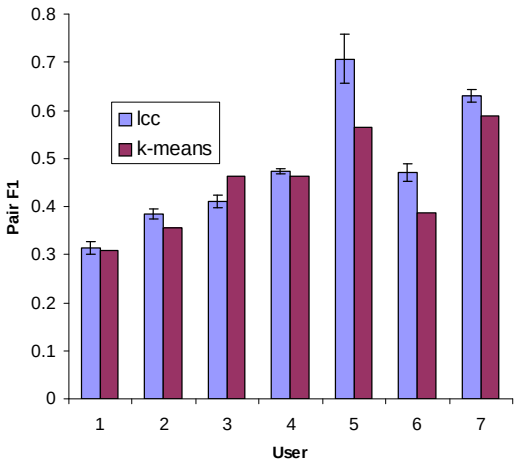
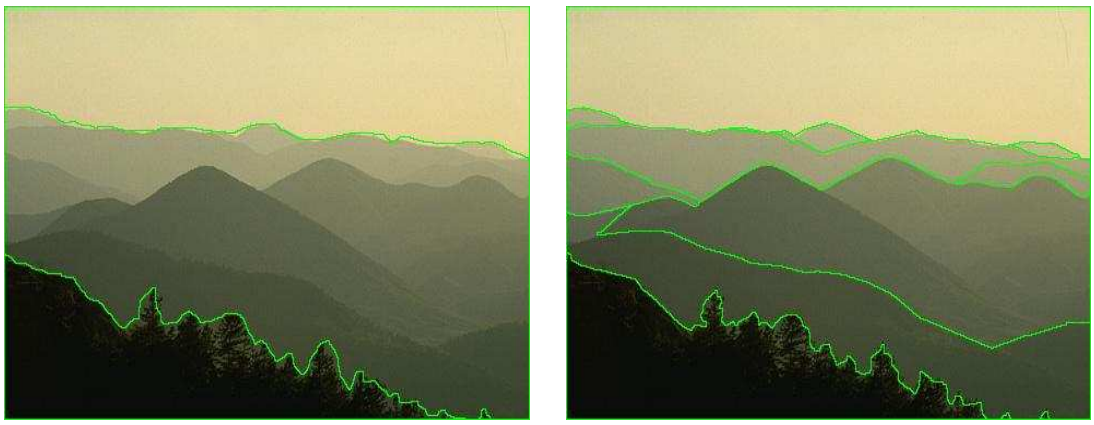


Figure 11.2: Average Pair F1 measure for clustering user mailboxes.

features, and took the best clustering out of five tries. We made this comparison somewhat easy for k-means by giving it the correct number of clusters k. We also informed our algorithm of the number of clusters by uniformly adding a positive weight to the edge weights to cause it to give the correct number of clusters. We performed a simple binary search on this additional bias weight parameter to find the number of clusters comparable to k. The results in Fig. 11.2 show the average $F1$ measure (harmonic mean of precision and recall; a standard metric in information retrieval [Baeza-Yates & Ribeiro-Neto, 1999]) computed on the pairs of messages that belonged to the same cluster. Our algorithm significantly outperforms k-means on several users and does worse only for one of the users.

11.4.3 Image segmentation

We also test our approach on the task of image segmentation. We selected images from the Berkeley Image Segmentation Dataset [Martin *et al.*, 2001] for which two users had significantly different segmentations. For example, Fig. 11.3(a) and (b) show two distinct segmentations: one very coarse, mostly based of overall hue, and one much finer, based on the hue and intensity. Depending on the task at hand, we may prefer the first over the second or vice-versa. It is precisely this kind of variability in the similarity judgements that



(a) (b)

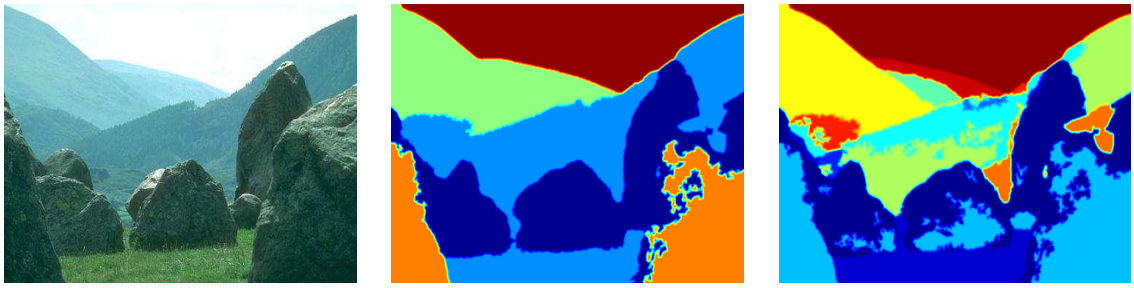
Figure 11.3: Two segmentations by different users: training image with (a) coarse segmentation and (b) fine segmentation.

we want our algorithm to capture.

In order to segment the image, we first divided it contiguous regions of approximately the same color by running connected components on the pixels. We connected two adjacent pixels by an edge if their RGB value was the same at a coarse resolution (4 bits per each of the R,G,B channel). We then selected about a hundred largest regions, which covered 80-90% of the pixels. These regions are the objects that our algorithm learns to cluster. (We then use the learned metric to greedily assign the remaining small regions to the large adjoining regions.)

There is a rich space of possible features we can use in our models: for each pair of regions, we can consider their shape, color distribution, distance, presence of edges between them, etc. In our experiments, we used a fairly simple set of features that are very easy to compute. For each region, we calculated the bounding box, area and average color (averaging the pixels in the RGB space). We then computed three distances (one for each HSV channel), as well as the distance in pixels between the bounding boxes and the area of the smaller of the two regions. All features were normalized to have zero mean and variance 1.

We trained two models, one using Fig. 11.3(a) and the other using Fig. 11.3(b) and



(a) (b) (c)

Figure 11.4: Test image: (a) input; (b) segmentation based on coarse training data (c) segmentation based on fine training data.

tested on the image in Fig. 11.4(a). The results are shown in Fig. 11.4(b) and (c), respectively. Note that mountains, rocks and grass are segmented very coarsely based on hue in (b) while the segmentation in (c) is more detailed and sensitive to saturation and value of the colors.

11.5 Related work

The performance of most clustering algorithms depends critically on the distance metric that they are given for measuring the similarity or dissimilarity between different datapoints. Recently, a number of algorithms have been proposed for automatically learning distance metrics as a preprocessing step for clustering [Xing *et al.*, 2002; Bar-Hillel *et al.*, 2003]. In contrast to algorithms that learn a metric independently of the algorithm that will be used to cluster the data, we describe a formulation that tightly integrates metric learning with the clustering algorithm, tuning one to the other in a joint optimization. Thus, instead of using an externally-defined criterion for choosing the metric, we will instead seek to learn a good metric *for the clustering algorithm*. An example of work in a similar vein is the algorithm for learning a distance metric for spectral clustering [Bach & Jordan, 2003]. The clustering algorithm essentially uses an eigenvector decomposition of an appropriate matrix derived from the pairwise affinity matrix, which is more efficient than correlation clustering, for which we use LP or SDP formulations. However the objective in the learning

formulation proposed in Bach and Jordan [2003] is not convex and difficult to optimize.

11.6 Conclusion

We looked at correlation clustering, and how to learn the edge weights from example clusterings. Our approach ties together the inference and learning algorithm, and attempts to learn a good metric specifically for the clustering algorithm.. We showed results on a synthetic dataset, showcasing robustness to noise dimensions. Experiments on the CALO e-mail and image segmentation experiments show the potential of the algorithm on real-world data. The main limitation of the correlation clustering is scalability: the number of constraints ($|\mathcal{V}|^3$) in the LP relaxation and the size of the positive-semidefinite constraint in the SDP relaxation. It would be very interesting to explore constraint generation or similar approaches to speed up learning and inference. On the theoretical side, it would be interesting to work out a PAC-like bound for generalization of the learned score metric.

Part IV

Conclusions and future directions

Chapter 12

Conclusions and future directions

This thesis presents a novel statistical estimation framework for structured models based on the large margin principle underlying support vector machines. The framework results in several efficient learning formulations for complex prediction tasks. Fundamentally, we rely on the expressive power of convex optimization problems to compactly capture inference or solution optimality in structured models. Directly embedding this structure within the learning formulation produces compact convex problems for efficient estimation of very complex models. For some of these models, alternative estimation methods are intractable. We develop theoretical foundations for our approach and show a wide range of experimental applications, including handwriting recognition, 3D terrain classification, disulfide connectivity in protein structure prediction, hypertext categorization, natural language parsing, email organization and image segmentation.

12.1 Summary of contributions

We view a structured prediction model as a mapping from the space of inputs $\mathbf{x} \in \mathcal{X}$ to a discrete vector output $\mathbf{y} \in \mathcal{Y}$. Essentially, a model defines a compact, parameterized scoring function $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$ and prediction using the model reduces to finding the highest scoring output $\mathbf{y}$ given the input $\mathbf{x}$. Our class of models has the following linear form:

$$h_{\mathbf{w}}(\mathbf{x}) = \operatorname*{arg\,max}_{\mathbf{y}\,:\,\mathbf{g}(\mathbf{x},\mathbf{y}) \leq 0} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}),$$

where $\mathbf{w} \in \mathbb{R}^n$ is the vector of parameters of the model, constraints $\mathbf{g}(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^k$ define the space of feasible outputs $\mathbf{y}$ given the input $\mathbf{x}$ and basis functions $\mathbf{f}(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n$ represent salient features of the input/output pair. Although the space of outputs $\{\mathbf{y} : \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq 0\}$ is usually immense, we assume that the inference problem $\arg\max_{\mathbf{y}\,:\,\mathbf{g}(\mathbf{x},\mathbf{y})\leq 0} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$ can be solved (or closely approximated) by an efficient algorithm that exploits the structure of the constraints $\mathbf{g}$ and basis functions $\mathbf{f}$. This definition covers a broad range of models, from probabilistic models such as Markov networks and context free grammars to more unconventional models like weighted graph-cuts and matchings.

12.1.1 Structured maximum margin estimation

Given a sample $S = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^m$, we develop methods for finding parameters $\mathbf{w}$ such that:

$$\arg\max_{\mathbf{y} \in \mathcal{Y}^{(i)}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y}) \;\approx\; \mathbf{y}^{(i)}, \quad \forall i,$$

where $\mathcal{Y}^{(i)} = \{\mathbf{y} : \mathbf{g}(\mathbf{x}^{(i)}, \mathbf{y}) \leq 0\}$.

The naive formulation[1] uses $\sum_i |\mathcal{Y}^{(i)}|$ linear constraints, which is generally exponential in the number of variables in each $\mathbf{y}^{(i)}$.

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y}), \quad \forall i, \;\; \forall \mathbf{y} \in \mathcal{Y}^{(i)}. \end{aligned}$$

We propose two general approaches that transform the above exponential size QP to an exactly equivalent polynomial size QP in many important classes of models. These formulations allow us to find globally optimal parameters (with fixed precision) in polynomial time using standard optimization software. In many models where maximum likelihood estimation is intractable, we provide exact maximum margin solutions (Ch. 7 and Ch. 10).

[1] For simplicity, we omit the slack variables in this summary.

Min-max formulation

We can turn the above problem into an equivalent min-max formulation with i non-linear max-constraints:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) \geq \max_{\mathbf{y} \in \mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})], \quad \forall i. \end{aligned}$$

The key to solving the estimation problem above efficiently is the loss-augmented inference problem $\max_{\mathbf{y} \in \mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$. Even if $\max_{\mathbf{y} \in \mathcal{Y}^{(i)}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$ can be solved in polynomial time using convex optimization, the form of the loss term $\ell_i(\mathbf{y})$ is crucial for the loss-augmented inference to remain tractable. We typically use a natural loss function which is essentially the Hamming distance between $\mathbf{y}^{(i)}$ and $h(\mathbf{x}^{(i)})$: the number of variables predicted incorrectly.

We show that if we can express the (loss-augmented) inference as a compact convex optimization problem (e.g., LP, QP, SDP, etc.), we can embed the maximization inside the min-max formulation to get a compact convex program equivalent to the naive exponential formulation. We show that this approach leads to exact polynomial-size formulations for estimation of low-treewidth Markov networks, associative Markov networks over binary variables, context-free grammars, bipartite matchings, and many other models.

Certificate formulation

There are several important combinatorial problems which allow polynomial time solution yet do not have a compact convex optimization formulation. For example, maximum weight perfect (non-bipartite) matching and spanning tree problems can be expressed as linear programs with *exponentially* many constraints, but no polynomial formulation is known [Bertsimas & Tsitsiklis, 1997; Schrijver, 2003]. Both of these problems, however, can be solved in polynomial time using combinatorial algorithms. In some cases, though, we can find a compact *certificate of optimality* that guarantees that

$$\mathbf{y}^{(i)} = \arg\max_{\mathbf{y}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})].$$

For perfect (non-bipartite) matchings, this certificate is a condition that ensures there are no augmenting alternating cycles (see Ch. 10). We can express this condition by defining an auxiliary distance function on the nodes an a set of linear constraints that are satisfied if and only if there are no negative cycles. This simple set of linear constraints scales linearly with the number of edges in the graph. Similarly, we can derive a compact certificate for the spanning tree problem.

The certificate formulation relies on the fact that verifying optimality of a solution is often easier than actually finding one. This observation allows us to apply our framework to an even broader range of models with combinatorial structure than the min-max formulation.

Maximum margin vs. maximum likelihood

There are several theoretical advantages to our approach in addition to the empirical accuracy improvements we have shown experimentally. Because our approach only relies on using the maximum in the model for prediction, and does not require a normalized distribution $P(\mathbf{y} \mid \mathbf{x})$ over all outputs, maximum margin estimation can be tractable when maximum likelihood is not. For example, to learn a probabilistic model $P(\mathbf{y} \mid \mathbf{x})$ over bipartite matchings using maximum likelihood requires computing the normalizing partition function, which is #P-complete [Valiant, 1979; Garey & Johnson, 1979]. By contrast, maximum margin estimation can be formulated as a compact QP with linear constraints. Similar results hold for an important subclass of Markov networks and non-bipartite matchings.

In models that are tractable for both maximum likelihood and maximum margin (such as low-treewidth Markov networks, context free grammars, many other problems in which inference is solvable by dynamic programming), our approach has an additional advantage. Because of the hinge-loss, the solutions to the estimation are relatively sparse in the dual space (as in SVMs), which makes the use of kernels much more efficient. Maximum likelihood models with kernels are generally non-sparse and require pruning or greedy support vector selection methods [Wahba *et al.*, 1993; Zhu & Hastie, 2001; Lafferty *et al.*, 2004; Altun *et al.*, 2004].

There are, of course, several advantages to maximum likelihood estimation. In applications where probabilistic confidence information is a must, maximum likelihood is much more appropriate. Also, in training settings with missing data and hidden variables, probabilistic interpretation permits the use of well-understood algorithms such as EM [Dempster *et al.*, 1977].

Approximations

In many problems, the maximization problem we are interested in may be NP-hard, for example, we consider MAP inference in large treewidth Markov networks in Ch. 8, multi-way cuts in Ch. 7, graph-partitioning in Ch. 11. Many such problems can be written as *integer* programs. Relaxations of such integer programs into LPs, QPs or SDPs often provide excellent approximation algorithms and fit well within our framework, particularly the min-max formulation. We show empirically that these approximations are very effective in many applications.

12.1.2 Markov networks: max-margin, associative, relational

The largest portion of the thesis is devoted to novel estimation algorithms, representational extensions, generalization analysis and experimental validation for Markov networks.

- **Low-treewidth Markov networks**
  We use a compact LP for MAP inference in Markov networks with sequence and other low-treewidth structure to derive an exact, compact, convex learning formulation. The dual formulation allows efficient integration of kernels with graphical models that leverages rich high-dimensional representations for accurate prediction in real-world tasks.
- **Scalable online algorithm**
  Although our convex formulation is a QP with linear number of variables and constraints in the size of the data, for large datasets (millions of examples), it is very difficult to solve using standard software. We present an efficient algorithm for solving the estimation problem called Structured SMO. Our online-style algorithm uses

inference in the model and analytic updates to solve extremely large estimation problems.

- **Generalization analysis**
  We analyze the theoretical generalization properties of max-margin estimation in Markov networks and derive a novel margin-based bound for structured prediction. This is the first bound to address structured error (e.g., proportion of mislabeled pixels in an image).

- **Learning associative Markov networks (AMNs)**
  We define an important subclass of Markov networks that captures positive correlations present in many domains. This class of networks extends the Potts model [Potts, 1952] often used in computer vision and allows exact MAP inference in the case of binary variables. We show how to express the inference problem using an LP which is exact for binary networks. As a result, for associative Markov networks over binary variables, our framework allows exact estimation of networks of arbitrary connectivity and topology, for which likelihood methods are believed to be intractable. For the non-binary case, we provide an approximation that works well in practice. We present an AMN-based method for object segmentation from 3D range data. By constraining the class of Markov networks to AMNs, our models are learned efficiently and, at run-time, can scale up to tens of millions of nodes and edges by using graph-cut based inference [Kolmogorov & Zabih, 2002].

- **Representation and learning of relational Markov networks**
  We introduce relational Markov networks (RMNs), which compactly define templates for Markov networks for domains with relational structure: objects, attributes, relations. The graphical structure of an RMN is based on the relational structure of the domain, and can easily model complex interaction patterns over related entities. We apply this class of models to classification of hypertext using hyperlink structure to define relations between webpages. We use a compact approximate MAP LP in these complex Markov networks, in which exact inference is intractable, to derive an approximate max-margin formulation.

12.1.3 Broader applications: parsing, matching, clustering

The other large portion of the thesis addresses a range of prediction tasks with very diverse models: context free grammars for natural language parsing, perfect matchings for disulfide connectivity in protein structure prediction, graph partitions for clustering documents and segmenting images.

- **Learning to parse**
  We exploit dynamic programming decomposition of context free grammars to derive a compact max-margin formulation. We build on a recently proposed "unlexicalized" grammar that allows cubic time parsing and we show how to achieve high-accuracy parsing (still in cubic time) by exploiting novel kinds of lexical information. We show experimental evidence of the model's improved performance over several baseline models.

- **Learning to match**
  We use a combinatorial optimality condition, namely the absence of augmenting alternating cycles, to derive an exact, efficient certificate formulation for learning to match. We apply our framework to prediction of disulfide connectivity in proteins using perfect matchings. The algorithm we propose uses kernels, which makes it possible to efficiently embed input features in very high-dimensional spaces and achieve state-of-the-art accuracy.

- **Learning to cluster**
  By expressing the correlation clustering problem as a compact LP and SDP, we use the min-max formulation to learn a parameterized scoring function for clustering. In contrast to algorithms that learn a metric independently of the algorithm that will be used to cluster the data, we describe a formulation that tightly integrates metric learning with the clustering algorithm, tuning one to the other in a joint optimization. We formulate the approximate learning problem as a compact convex program. Experiments on synthetic and real-world data show the ability of the algorithm to learn an appropriate clustering metric for a variety of desired clusterings, including email folder organization and image segmentation.

12.2 Extensions and open problems

There are several immediate applications, less immediate extensions and open problems for our estimation framework. We organize these ideas into several sections below, including further theoretical analysis and new optimization algorithms, novel prediction tasks, and more general learning settings.

12.2.1 Theoretical analysis and optimization algorithms

- **Approximation bounds**
  In several of the intractable models, like multi-class AMNs in Ch. 7 and correlation clustering in Ch. 11, we used approximate convex programs within the min-max formulation. These approximate inference programs have strong relative error bounds. An open question is to translate these error bounds on inference into error bounds on the resulting max-margin formulations.

- **Generalization bounds with distributional assumptions**
  In Ch. 5, we presented a bound on the structured error in Markov networks, without any assumption about the distribution of $P(\mathbf{y} \mid \mathbf{x})$, relying only on the samples $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ being i.i.d. This distribution-free assumption leads to a worst case analysis, while some assumptions about the approximate decomposition $P(\mathbf{y} \mid \mathbf{x})$ may be warranted. For example, for sequential prediction problems, the Markov assumption of some finite order is reasonable (i.e., given the input and previous k labels, the next label is independent of the labels more than k in the past). In spatial prediction tasks, a label variable is independent of the rest given a large enough ball of labels around it. Similar assumptions may be made for some "degree of separation" in relational domains. More generally, it would be interesting to exploit such conditional independence assumptions or asymptotic bounds on entropy of $P(\mathbf{y} \mid \mathbf{x})$ to get generalization guarantees even from a single structured example $(\mathbf{x}, \mathbf{y})$.

- **Problem-specific optimization methods**
  Although our convex formulations are polynomial in the size of the data, scaling

up to larger datasets will require problem-specific optimization methods. For low-treewidth Markov networks and context free grammars, we have presented the Structured SMO algorithm. Another algorithm useful for such models is Exponentiated Gradient [Bartlett *et al.*, 2004]. Both algorithms rely on dynamic programming decompositions. However, models which do not permit such decompositions, such as graph-cuts, matchings, and many others, create a need for new algorithms that can effectively use combinatorial optimization as a subroutine to eliminate the dependence on general-purpose convex solvers.

12.2.2 Novel prediction tasks

- **Bipartite matchings**
  Maximum weight bipartite matchings are used in a variety of problems to predict mappings between sets of items. In machine translation, matchings are used to map words of the two languages in aligned sentences [Matusov *et al.*, 2004]. In 2D shape matching, points on two shapes are matched based on their local contour features [Belongie *et al.*, 2002]. Our framework provides an exact, efficient alternative to the maximum likelihood estimation for learning the matching scoring function.

- **Sequence alignment**
  In standard pairwise alignment of biological sequences, a string edit distance is used to determine which portions of the sequences align to each other [Needleman & Wunsch, 1970; Durbin *et al.*, 1998]. Finding the best alignment involves a dynamic program that generalizes the longest common subsequence algorithm. Our framework can be applied (just as in context free grammar estimation) to efficiently learn a more complex edit function that depends on the contextual string features, perhaps using novel string kernels [Haussler, 1999; Leslie *et al.*, 2002; Lodhi *et al.*, 2000].

- **Continuous prediction problems**
  We have addressed estimation of models with discrete output spaces, generalizing classification models to multivariate, structured classification. Similarly, we can consider a whole range of problems where the prediction variables are continuous.

Such problems are a natural generalizations of regression, involving correlated, inter-constrained real-valued outputs. For example, several recent models of metabolic flux in yeast use linear programming formulations involving quantities of various enzymes, with stoichiometric constraints [Varma & Palsson, 1994]. It would be interesting to use observed equilibria data under different conditions to learn what "objective" the cell is maximizing. In financial modeling, convex programs are often used to model portfolio management; for example, Markowitz portfolio optimization is formulated as a quadratic program which minimizes risk and maximizes expected return under budget constraints [Markowitz, 1991; Luenberger, 1997]. In this setting, one could learn a user's return projection and risk assessment function from observed portfolio allocations by the user.

These problems are similar to the discrete structured prediction models we have considered: inference in the model can formulated as a convex optimization problem. However, there are obstacles to directly applying the min-max or certificate formulations. Details of this are beyond the scope of this thesis, but it suffices to say that loss-augmented inference using, Hamming distance equivalent, L_1 loss (or L_2 loss), no longer produces a maximization of a concave objective with convex constraints since L_1, L_2 are convex, not concave (it turns out that it is actually possible to use L_∞ loss). Developing effective loss functions and max-margin formulations for the continuous setting could provide a novel set of effective models for structured multivariate real-valued prediction problems.

12.2.3 More general learning settings

- **Structure learning**

  We have focused on the problem of learning parameters of the model (even though our kernelized models can be considered non-parametric). In the case of Markov networks, especially in spatial and relational domains, there is a wealth of possible structures (cliques in the network) one can use to model a problem. It is particularly interesting to explore the problem of inducing these cliques automatically from data. The standard method of greedy stepwise selection followed by re-estimation is very

expensive in general networks [Della Pietra *et al.*, 1997; Bach & Jordan, 2001]. Recent work on selecting input features in Markov networks (or CRFs) uses several approximations to learn efficiently with millions of candidate features [McCallum, 2003]. However, clique selection is still relatively unexplored. It is possible that AMNs, by restricting the network to be tractable under any structure, may permit more efficient clique selection methods.

- **Semi-supervised learning**

  Throughout the thesis we have assumed completely labeled data. This assumption often limits us to relatively small training sets where data has been carefully annotated, while much of the easily accessible data is not at all or suitably labeled. There are several more general settings we would like to extend our framework.

  The simplest setting is a mix of labeled and unlabeled examples, where a small supervised dataset is augmented by a large unsupervised one. There has been much research in this setting for classification [Blum & Mitchell, 1998; Nigam *et al.*, 2000; Chapelle *et al.*, 2002; Szummer & Jaakkola, 2001; Zhu *et al.*, 2003; Corduneanu & Jaakkola, 2003]. Although most of this work has been done in a probabilistic setting, the principle of regularizing (discouraging) decision boundaries near densely clustered inputs could be applicable to our structured setting.

  A more complex and rich setting involves presence of hidden variables in each example. For example, in machine translation, word correspondences between pairs of sentences are usually not manually annotated (at least not on a large scale). These correspondence variables can be treated as hidden variables. Similarly, in handwriting recognition, we may not have each letter segmented out but instead just get a word or sentence as a label for the entire image. This setting has been studied mainly in the probabilistic, generative models often using the EM algorithm [Dempster *et al.*, 1977; Cowell *et al.*, 1999]. Discriminative methods have been explored far less. Especially in the case of combinatorial structures, extensions of our framework allow opportunities for problem-specific convex approximations to be exploited.

12.3 Future

We have presented a supervised learning framework for a large class of prediction models with rich and interesting structure. Our approach has several theoretical and practical advantages over standard probabilistic models and estimation methods for structured prediction. We hope that continued research in this framework will help tackle evermore sophisticated prediction problems in the future.

Appendix A

Proofs and derivations

A.1 Proof of Theorem 5.5.1

The proof of Theorem 5.5.1 uses the covering number bounds of Zhang [2002] (in the Data-Dependent Structural Risk Minimization framework [Shawe-Taylor *et al.*, 1998].) Zhang provides generalization guarantees for linear binary classifiers of the form $h_{\mathbf{w}}(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}^\top \mathbf{x})$. His analysis is based on the upper bounds on the covering number for the class of linear functions $\mathcal{F}_L(\mathbf{w}, \mathbf{z}) = \mathbf{w}^\top \mathbf{z}$ where the norms of the vectors $\mathbf{w}$ and $\mathbf{z}$ are bounded. We reproduce the relevant definitions and theorems from Zhang [2002] here to highlight the necessary extensions for structured classification.

The covering number is a key quantity in measuring function complexity. Intuitively, the covering number of an infinite class of functions (e.g. parameterized by a set of weights $\mathbf{w}$) is the number of vectors necessary to approximate the values of any function in the class on a sample. Margin-based analysis of generalization error uses the margin achieved by a classifier on the training set to approximate the original function class of the classifier by a finite covering with precision that depends on the margin. Here, we will only define the ∞-norm covering number.

A.1.1 Binary classification

In binary classification, we are given a sample $S = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^m$, from distribution D over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Y}$ is mapped to ± 1, so we can fold $\mathbf{x}$ and y into $\mathbf{z} = y\mathbf{x}$.

Definition A.1.1 (Covering Number) *Let $\nu = \{\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(r)}\}$, where $\mathbf{v}^{(j)} \in \mathbb{R}^m$, be a* covering *of a function class $\mathcal{F}(\mathbf{w}, S)$ with ϵ-precision under the metric ρ, if for all $\mathbf{w}$ there exists a $\mathbf{v}^{(j)}$ such that for each data sample $\mathbf{z}^{(i)} \in S$:*

$$\rho(\mathbf{v}_i^{(j)}, \mathcal{F}(\mathbf{w}, \mathbf{z}^{(i)})) \leq \epsilon.$$

The covering number of a sample S *is the size of the smallest covering: $\mathcal{N}_\infty(\mathcal{F}, \rho, \epsilon, S) = \inf |\nu|$ s.t. ν is a covering of $\mathcal{F}(\mathbf{w}, S)$. We also define the* covering number for any sample of size m*: $\mathcal{N}_\infty(\mathcal{F}, \rho, \epsilon, m) = \sup_{S:\ |S|=m} \mathcal{N}_\infty(\mathcal{F}, \rho, \epsilon, S)$.* ∎

When the norms of $\mathbf{w}$ and $\mathbf{z}$ are bounded, we have the following upper bound on the covering number of linear functions under the linear metric $\rho_L(v, v') = |v - v'|$.

Theorem A.1.2 (Theorem 4 from Zhang [2002]) *If $\|\mathbf{w}\|_2 \leq a$ and $\|\mathbf{z}\|_2 \leq b$, then $\forall$ $\epsilon > 0$,*

$$\log_2 \mathcal{N}_\infty(\mathcal{F}_L, \rho_L, \epsilon, m) \leq 36 \frac{a^2 b^2}{\epsilon^2} \log_2 \left(2 \lceil 4ab/\epsilon + 2 \rceil m + 1\right). \ ∎$$

In order to use the classifier's margin to bound its expected loss, the bounds below use a stricter, margin-based loss on the training sample that measures the worst loss achieved by the approximate covering based on this margin. Let $f : \mathbb{R} \mapsto [0, 1]$ be a loss function. In binary classification, we let $f(v) = \mathbb{1}(v \leq 0)$ be the step function, so that 0-1 loss of $\operatorname{sgn}(\mathbf{w}^\top \mathbf{x})$ is $f(\mathcal{F}_L(\mathbf{w}, \mathbf{z}))$. The next theorem bounds the expected f loss in terms of the γ-margin loss, $f^\gamma(v) = \sup_{\rho(v,v') < 2\gamma} f(v')$, on the training sample. For 0-1 loss and linear metric ρ_L, the corresponding γ-margin loss is $f^\gamma(v) = \mathbb{1}(v \leq 2\gamma)$.

Theorem A.1.3 (Corollary 1 from Zhang [2002]) *Let $f : \mathbb{R} \mapsto [0, 1]$ be a loss function and $f^\gamma(v) = \sup_{\rho(v,v') < 2\gamma} f(v')$ be the γ-margin loss for a metric ρ. Let $\gamma_1 > \gamma_2 > \ldots$ be*

a decreasing sequence of parameters, and p_i be a sequence of positive numbers such that $\sum_{i=1}^{\infty} p_i = 1$, then for all $\delta > 0$, with probability of at least $1 - \delta$ over data:

$$\mathbf{E}_D[f(\mathcal{F}(\mathbf{w}, \mathbf{z}))] \quad \leq \quad \mathbf{E}_S[f^{\gamma}(\mathcal{F}(\mathbf{w}, \mathbf{z}))] + \sqrt{\frac{32}{m}\left[\ln 4\mathcal{N}_{\infty}(\mathcal{F}, \rho, \gamma_i, S) + \ln \frac{1}{p_i \delta}\right]}$$

for all $\mathbf{w}$ and γ, where for each fixed γ, we use i to denote the smallest index s.t. $\gamma_i \leq \gamma$.

A.1.2 Structured classification

We will extend this framework to bound the average per-label loss $\ell^H(\mathbf{y})/L$ for structured classification by defining an appropriate loss f and a function class $\mathcal{F}$ (as well as a metric ρ) such that $f(\mathcal{F})$ computes average per-label loss and $f^{\gamma}(\mathcal{F})$ provides a suitable γ-margin loss. We will bound the corresponding covering number by building on the bound in Theorem A.1.2.

We can no longer simply fold $\mathbf{x}$ and $\mathbf{y}$, since $\mathbf{y}$ is a vector, so we let $\mathbf{z} = (\mathbf{x}, \mathbf{y})$. In order for our loss function to compute average per-label loss, it is convenient to make our function class *vector-valued* (instead of scalar-valued as above). We define a new function class $\mathcal{F}_M(\mathbf{w}, \mathbf{z})$, which is a vector of minimum values of $\mathbf{w}^{\top}\Delta\mathbf{f}_i(\mathbf{y})$ for each error level $\ell^H(\mathbf{y})$ from 1 to L as described below.

Definition A.1.4 (dth-error-level function) *The* dth-error-level function $M_d(\mathbf{w}, \mathbf{z})$ *for* $d \in \{1, \ldots, L\}$ *is given by:*

$$M_d(\mathbf{w}, \mathbf{z}) = \min_{\mathbf{y}:\ell^H(\mathbf{y})=d} \mathbf{w}^{\top}\Delta\mathbf{f}_i(\mathbf{y}). \quad \blacksquare$$

Definition A.1.5 (Multi-error-level function class) *The* multi-error-level function class $\mathcal{F}_M(\mathbf{w}, \mathbf{z})$ *is given by:*

$$\mathcal{F}_M(\mathbf{w}, \mathbf{z}) = (M_1(\mathbf{w}, \mathbf{z}), \ldots, M_d(\mathbf{w}, \mathbf{z}), \ldots, M_L(\mathbf{w}, \mathbf{z}))\,. \quad \blacksquare$$

We can now compute the average per-label loss from $\mathcal{F}_M(\mathbf{w}, \mathbf{z})$ by defining an appropriate loss function f_M.

Definition A.1.6 (Average per-label loss) *The* average per-label loss $f_M : I\!R^{\,L} \mapsto [0,1]$ *is given by:*

$$f_M(\mathbf{v}) = \frac{1}{L} \arg \min_{d:v_d \leq 0} v_d,$$

where in case $\forall d, \quad v_d > 0$*, we define* $\arg\min_{d:v_d\leq 0} v_d \equiv 0$. ∎

With the above definitions, we have an upper bound on the average per-label loss

$$f_M(\mathcal{F}_M(\mathbf{w},\mathbf{z})) = \frac{1}{L} \arg \min_{d:M_d(\mathbf{w},\mathbf{z})\leq 0} M_d(\mathbf{w},\mathbf{z}) \geq \frac{1}{L}\ell^H\big(\arg\max_{\mathbf{y}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})\big).$$

Note that the case $\forall d, \quad M_d(\mathbf{w},\mathbf{z}) > 0$ corresponds to the classifier making no mistakes: $\arg\max_{\mathbf{y}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) = \mathbf{y}$. This upper bound is tight if $\mathbf{y} = \arg\max_{\mathbf{y}'} \mathbf{w}^\top \mathbf{f}(\mathbf{x},\mathbf{y}')$, Otherwise, it is adversarial: it picks from all $\mathbf{y}'$ which are better ($\mathbf{w}^\top \mathbf{f}(\mathbf{y}) \leq \mathbf{w}^\top \mathbf{f}(\mathbf{y}')$), one that maximizes the Hamming distance from $\mathbf{y}$.

We now need to define an appropriate metric ρ that in turn defines γ-margin loss for structured classification. Since the margin of the hypothesis grows with the number of mistakes, our metric can become "looser" with the number of mistakes, as there is more room for error.

Definition A.1.7 (Multi-error-level metric) *Let the* multi-error-level metric $\rho_M : I\!R^L \times I\!R^L \mapsto I\!R$ *for a vector in* $I\!R^{\,L}$ *be given by:*

$$\rho_M(\mathbf{v},\mathbf{v}') = \max_d \frac{|v_d - v'_d|}{d}. \quad ∎$$

We now define the corresponding γ-margin loss using the new metric:

Definition A.1.8 (γ-margin average per-label loss) *The* γ-margin average per-label loss $f_M^\gamma : I\!R^{\,L} \mapsto [0,1]$ *is given by:*

$$f_M^\gamma(\mathbf{v}) = \sup_{\rho_M(\mathbf{v},\mathbf{v}')\leq 2\gamma} f_M(\mathbf{v}'). \quad ∎$$

Combining the two definitions, we get:

$$f_M^\gamma(\mathcal{F}_M(\mathbf{w},\mathbf{z})) \quad = \quad \sup_{\mathbf{v}:|v_d - M_d(\mathbf{w},\mathbf{z})|\leq 2d\gamma} \frac{1}{L} \arg \min_{d:v_d\leq 0} v_d.$$

We also define the corresponding covering number for our vector-valued function class:

Definition A.1.9 (Multi-error-level covering number) *Let $\mathcal{V} = \{\mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(r)}\}$, where $\mathbf{V}^{(j)} = (\mathbf{V}_1^{(j)}, \ldots, \mathbf{V}_i^{(j)}, \ldots, \mathbf{V}_m^{(j)})$ and $\mathbf{V}_i^{(j)} \in I\!R^L$, be a* covering *of $\mathcal{F}_M(\mathbf{w}, S)$, with ϵ-precision under the metric ρ_M, if for all $\mathbf{w}$ there exists a $\mathbf{V}^{(j)}$ such that for each data sample $\mathbf{z}^{(i)} \in S$:*

$$\rho_M(\mathbf{V}_i^{(j)}, \mathcal{F}_M(\mathbf{w}, \mathbf{z}^{(i)})) \leq \epsilon.$$

The covering number of a sample S *is the size of the smallest covering: $\mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon, S) = \inf |\mathcal{V}|$ s.t. $\mathcal{V}$ is a covering of $\mathcal{F}_M(\mathbf{w}, S)$. We also define*

$$\mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon, m) = \sup_{S:\ |S|=m} \mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon, S).$$

We provide a bound on the covering number of our new function class in terms of a covering number for the linear function class. Recall that N_c is the maximum number of cliques in $\mathcal{G}(\mathbf{x})$, V_c is the maximum number of values in a clique $|\mathbf{Y}_c|$, q is the maximum number of cliques that have a variable in common, and R_c is an upper-bound on the 2-norm of clique basis functions. Consider a first-order sequence model as an example, with L as the maximum length, and V the number of values a variable takes. Then $N_c = 2L - 1$ since we have L node cliques and $L - 1$ edge cliques; $V_c = V^2$ because of the edge cliques; and $q = 3$ since nodes in the middle of the sequence participate in 3 cliques: previous-current edge clique, node clique, and current-next edge clique.

Lemma A.1.10 (Bound on multi-error-level covering number)

$$\mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon q, m) \leq \mathcal{N}_\infty(\mathcal{F}_L, \rho_L, \epsilon, mN_c(V_c - 1)).$$

Proof: *We will show that $\mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon q, S) \leq \mathcal{N}_\infty(\mathcal{F}_L, \rho_L, \epsilon, S')$ for any sample S of size m, where we construct the sample S' of size $mN_c(V_c - 1)$ in order to cover the clique potentials as described below. Note that this is sufficient since $\mathcal{N}_\infty(\mathcal{F}_L, \rho_L, \epsilon, S') \leq \mathcal{N}_\infty(\mathcal{F}_L, \rho_L, \epsilon, mN_c(V_c - 1))$, by definition, so*

$$\mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon q, m) = \sup_{S:|S|=m} \mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon q, S) \leq \mathcal{N}_\infty(\mathcal{F}_L, \rho_L, \epsilon, mN_c(V_c - 1)).$$

The construction of S' below is inspired by the proof technique in Collins [2001], but the key difference is that our construction is linear in the number of cliques N_c and exponential in the number of label variables per clique, while his is exponential in the total number of label variables per example. This reduction in size comes about because our covering approximates the values of clique potentials $\mathbf{w}^\top \Delta\mathbf{f}_{i,c}(\mathbf{y}_c)$ for each clique c and clique assignment $\mathbf{y}_c$ as opposed to the values of entire assignments $\mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y})$.

For each sample $\mathbf{z} \in S$, we create $N_c(V_c - 1)$ samples $\Delta\mathbf{f}_{i,c}(\mathbf{y}_c)$, one for each clique c and each assignment $\mathbf{y}_c \neq \mathbf{y}_c^{(i)}$. We construct a set of vectors $\nu = \{\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(r)}\}$, where $\mathbf{v}^{(j)} \in I\!R^{mN_c(V_c-1)}$. The component of $\mathbf{v}^{(j)}$ corresponding to the sample $\mathbf{z}^{(i)}$ and the assignment $\mathbf{y}_c$ to the labels of the clique c will be denoted by $\mathbf{v}_{i,c}^{(j)}(\mathbf{y}_c)$. For convenience, we define $\mathbf{v}_{i,c}^{(j)}(\mathbf{y}_c^{(i)}) = 0$ for correct label assignments, as $\Delta\mathbf{f}_{i,c}(\mathbf{y}_c^{(i)}) = 0$. To make ν an ∞-norm covering of $\mathcal{F}_L(\mathbf{w}, S')$ under ρ_L, we require that for any $\mathbf{w}$ there exists a $\mathbf{v}^{(j)} \in \nu$ such that for each sample $\mathbf{z}^{(i)}$:

$$|\mathbf{v}_{i,c}^{(j)}(\mathbf{y}_c) - \mathbf{w}^\top \Delta\mathbf{f}_{i,c}(\mathbf{y}_c)| \leq \epsilon; \;\; \forall c \in \mathcal{C}^{(i)}, \; \forall \mathbf{y}_c. \tag{A.1}$$

By Definition A.1.1, the number of vectors in ν is given by $r = \mathcal{N}_\infty(\mathcal{F}_L, \rho_L, \epsilon, mN_c(V_c-1))$.

We can now use ν to construct a covering $\mathcal{V} = \{\mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(r)}\}$, where

$$\mathbf{V}^{(j)} = (\mathbf{V}_1^{(j)}, \ldots, \mathbf{V}_i^{(j)}, \ldots, \mathbf{V}_m^{(j)})$$

and $\mathbf{V}_i^{(j)} \in I\!R^L$, for our multi-error-level function $\mathcal{F}_M$. Let $\mathbf{v}_i^{(j)}(\mathbf{y}) = \sum_c \mathbf{v}_{i,c}^{(j)}(\mathbf{y}_c)$, and $M_d(\mathbf{v}_i^{(j)}, \mathbf{z}^{(i)}) = \min_{\mathbf{y}:\ell_i^H(\mathbf{y})=d} \mathbf{v}_i^{(j)}(\mathbf{y})$, then

$$\mathbf{V}_i^{(j)} = (M_1(\mathbf{v}^{(j)}, \mathbf{z}^{(i)}), \ldots, M_d(\mathbf{v}^{(j)}, \mathbf{z}^{(i)}), \ldots, M_L(\mathbf{v}^{(j)}, \mathbf{z}^{(i)}))\,. \tag{A.2}$$

Note that $\mathbf{v}_{i,c}^{(j)}(\mathbf{y}_c)$ is zero for all cliques c for which the assignment is correct: $\mathbf{y}_c = \mathbf{y}_c^{(i)}$. Thus for an assignment $\mathbf{y}$ with d mistakes, at most dq $\mathbf{v}_{i,c}^{(j)}(\mathbf{y}_c)$ will be non-zero, as each label can appear in at most q cliques. By combining this fact with Eq. (A.1), we obtain:

$$\left|\mathbf{v}_i^{(j)}(\mathbf{y}) - \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y})\right| \leq dq\epsilon, \qquad \forall i, \;\; \forall \mathbf{y} : \ell_i^H(\mathbf{y}) = d. \tag{A.3}$$

We conclude the proof by showing that $\mathcal{V}$ is a covering of $\mathcal{F}_M$ under ρ_M: For each $\mathbf{w}$, pick $\mathbf{V}^{(j)} \in \mathcal{V}$ such that the corresponding $\mathbf{v}^{(j)} \in \nu$ satisfies the condition in Eq. (A.1). We must now bound:

$$\rho_M(\mathbf{V}_i^{(j)}, \mathcal{F}_M(\mathbf{w}, \mathbf{z}^{(i)})) = \max_d \frac{|\min_{\mathbf{y}:\ell_i^H(\mathbf{y})=d} \mathbf{v}_i^{(j)}(\mathbf{y}) - \min_{\mathbf{y}:\ell_i^H(\mathbf{y})=d} \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y})|}{d}.$$

Fix any i. Let $\mathbf{y}_d^{\mathbf{v}} = \arg\min_{\mathbf{y}:\ell_i^H(\mathbf{y})=d} \mathbf{v}_i^{(j)}(\mathbf{y})$ and $\mathbf{y}_d^{\mathbf{w}} = \arg\min_{\mathbf{y}:\ell_i^H(\mathbf{y})=d} \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y})$. Consider the case where $\mathbf{v}_i^{(j)}(\mathbf{y}_d^{\mathbf{v}}) \geq \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}_d^{\mathbf{w}})$ (the reverse case is analogous), we must prove that:

$$\mathbf{v}_i^{(j)}(\mathbf{y}_d^{\mathbf{v}}) - \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}_d^{\mathbf{w}}) \leq \mathbf{v}_i^{(j)}(\mathbf{y}_d^{\mathbf{w}}) - \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}_d^{\mathbf{w}}) \leq dq\epsilon\ ; \tag{A.4}$$

where the first step follows from definition of $\mathbf{y}_d^{\mathbf{v}}$, since $\mathbf{v}_i^{(j)}(\mathbf{y}_d^{\mathbf{v}}) \leq \mathbf{v}_i^{(j)}(\mathbf{y}_d^{\mathbf{w}})$. The last step is a direct consequence of Eq. (A.3). Hence $\rho_M(\mathbf{V}_i^{(j)}, \mathcal{F}_M(\mathbf{w}, \mathbf{z}^{(i)})) \leq q\epsilon$. ∎

Lemma A.1.11 (Numeric bound on multi-error-level covering number)

$$\log_2 \mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon, m) \leq 36 \frac{R_c^2 \|\mathbf{w}\|_2^2 q^2}{\epsilon^2} \log_2 \left(1 + 2\left\lceil 4\frac{R_c \|\mathbf{w}\|_2 q}{\epsilon} + 2\right\rceil m N_c (V_c - 1)\right).$$

Proof: *Substitute Theorem A.1.2 into Lemma A.1.10.* ∎

Theorem A.1.12 (Multi-label analog of Theorem A.1.3) *Let f_M and $f_M^\gamma(v)$ be as defined above. Let $\gamma_1 > \gamma_2 > \ldots$ be a decreasing sequence of parameters, and p_i be a sequence of positive numbers such that $\sum_{i=1}^\infty p_i = 1$, then for all $\delta > 0$, with probability of at least $1 - \delta$ over data:*

$$E_{\mathbf{z}} f_M(\mathcal{F}_M(\mathbf{w}, \mathbf{z})) \quad \leq \quad E_S f_M^\gamma(\mathcal{F}_M(\mathbf{w}, \mathbf{z})) + \sqrt{\frac{32}{m}\left[\ln 4\mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \gamma_i, S) + \ln \frac{1}{p_i \delta}\right]}$$

for all $\mathbf{w}$ and γ, where for each fixed γ, we use i to denote the smallest index s.t. $\gamma_i \leq \gamma$.
Proof: *Similar to the proof of Zhang's Theorem 2 and Corollary 1 [Zhang, 2002] where in Step 3 (derandomization) we substitute the vector-valued $\mathcal{F}_M$ and the metric ρ_M.* ∎

Theorem 5.5.1 follows from above theorem with $\gamma_i = R_c \|\mathbf{w}\|_2 / 2^i$ and $p_i = 1/2^i$ using an

argument identical to the proof of Theorem 6 in Zhang [2002].

A.2 AMN proofs and derivations

In this appendix, we present proofs of the LP inference properties and derivations of the factored primal and dual max-margin formulation from Ch. 7. Recall that the LP relaxation for finding the optimal $\max_{\mathbf{y}} g(\mathbf{y})$ is:

$$\begin{aligned} \max \quad & \sum_{v\in\mathcal{V}}\sum_{k=1}^{K}\mu_v(k)g_v(k) + \sum_{c\in\mathcal{C}\setminus\mathcal{V}}\sum_{k=1}^{K}\mu_c(k)g_c(k) \\ \text{s.t.} \quad & \mu_c(k)\geq 0, \quad \forall c\in\mathcal{C},\ k; \quad \sum_{k=1}^{K}\mu_v(k)=1, \quad \forall v\in\mathcal{V}; \\ & \mu_c(k)\leq\mu_v(k), \quad \forall c\in\mathcal{C}\setminus\mathcal{V},\ v\in c,\ k. \end{aligned} \tag{A.5}$$

A.2.1 Binary AMNs

Proof (For Theorem 7.2.1) Consider any fractional, feasible μ. We show that we can construct a new feasible assignment μ' which increases the objective (or leaves it unchanged) and furthermore has fewer fractional entries.

Since $g_c(k)\geq 0$, we can assume that $\mu_c(k)=\min_{v\in c}\mu_v(k)$; otherwise we could increase the objective by increasing $\mu_c(k)$. We construct an assignment μ' from μ by leaving integral values unchanged and uniformly shifting fractional values by λ:

$$\begin{aligned} \mu'_v(1) &= \mu_v(1)-\lambda\mathbb{1}(0<\mu_v(1)<1), & \mu'_v(2) &= \mu_v(2)+\lambda\mathbb{1}(0<\mu_v(2)<1), \\ \mu'_c(1) &= \mu_c(1)-\lambda\mathbb{1}(0<\mu_c(1)<1), & \mu'_c(2) &= \mu_c(2)+\lambda\mathbb{1}(0<\mu_c(2)<1). \end{aligned}$$

Now consider the smallest fractional $\mu_v(k)$, $\lambda(k)=\min_{v\,:\,\mu_v(k)>0} y_v(k)$ for $k=1,2$. Note that if $\lambda=\lambda(1)$ or $\lambda=-\lambda(2)$, μ' will have at least one more integral $\mu'_v(k)$ than μ. Thus if we can show that the update results in a feasible and better scoring assignment, we can apply it repeatedly to get an optimal integer solution. To show that μ' is feasible, we need $\mu'_v(1)+\mu'_v(2)=1$, $\mu'_v(k)\geq 0$ and $\mu'_c(k)=\min_{i\in c}\mu'_v(k)$.

First, we show that $\mu'_v(1) + \mu'_v(2) = 1$.

$$\begin{aligned}\mu'_v(1) + \mu'_v(2) &= \mu_v(1) - \lambda \mathbb{1}(0 < \mu_v(1) < 1) + \mu_v(2) + \lambda \mathbb{1}(0 < \mu_v(2) < 1) \\ &= \mu_v(1) + \mu_v(2) = 1.\end{aligned}$$

Above we used the fact that if $\mu_v(1)$ is fractional, so is $\mu_v(2)$, since $\mu_v(1) + \mu_v(2) = 1$.

To show that $\mu'_v(k) \geq 0$, we prove $\min_v \mu'_v(k) = 0$.

$$\begin{aligned}\min_v \mu'_v(k) &= \min_v \left[\mu_v(k) - (\min_{i:\mu_v(k)>0} \mu_v(k)) \mathbb{1}(0 < \mu_v(k) < 1)\right] \\ &= \min\left(\min_i \mu_v(k), \min_{i:\mu_v(k)>0} \left[\mu_v(k) - \min_{i:\mu_v(k)>0} \mu_v(k)\right]\right) = 0.\end{aligned}$$

Lastly, we show $\mu'_c(k) = \min_{i \in c} \mu'_v(k)$.

$$\begin{aligned}\mu'_c(1) &= \mu_c(1) - \lambda \mathbb{1}(0 < \mu_c(1) < 1) \\ &= (\min_{i \in c} \mu_v(1)) - \lambda \mathbb{1}(0 < \min_{i \in c} \mu_v(1) < 1) = \min_{i \in c} \mu'_v(1); \\ \mu'_c(2) &= \mu_c(2) + \lambda \mathbb{1}(0 < \mu_c(1) < 1) \\ &= (\min_{i \in c} \mu_v(2)) + \lambda \mathbb{1}(0 < \min_{i \in c} \mu_v(2) < 1) = \min_{i \in c} \mu'_v(2).\end{aligned}$$

We have established that the new μ' are feasible, and it remains to show that we can improve the objective. We can show that the change in the objective is always λD for some constant D that depends only on μ and g. This implies that one of the two cases, $\lambda = \lambda(1)$ or $\lambda = -\lambda(2)$, will necessarily increase the objective (or leave it unchanged). The change in the objective is:

$$\begin{aligned}&\sum_{v \in \mathcal{V}} \sum_{k=1,2} [\mu'_v(k) - \mu_v(k)] g_v(k) + \sum_{c \in \mathcal{C} \setminus \mathcal{V}} \sum_{k=1,2} [\mu'_c(k) - \mu_c(k)] g_c(k) \\ &= \lambda \left[\sum_{v \in \mathcal{V}} [D_v(1) - D_v(2)] + \sum_{c \in \mathcal{C} \setminus \mathcal{V}} [D_c(1) - D_c(2)]\right] = \lambda D \\ &D_v(k) = g_v(k) \mathbb{1}(0 < \mu_v(k) < 1), \quad D_c(k) = g_c(k) \mathbb{1}(0 < \mu_c(k) < 1).\end{aligned}$$

Hence the new assignment μ' is feasible, does not decrease the objective function, and

has strictly fewer fractional entries. ∎

A.2.2 Multi-class AMNs

For $K > 2$, we use the randomized rounding procedure of Kleinberg and Tardos [1999] to produce an integer solution for the linear relaxation, losing at most a factor of $m = \max_{c \in \mathcal{C}} |c|$ in the objective function. The basic idea of the rounding procedure is to treat $\mu_v(k)$ as probabilities and assign labels according to these probabilities in phases. In each phase, we pick a label k, uniformly at random, and a threshold $\alpha \in [0, 1]$ uniformly at random. For each node i which has not yet been assigned a label, we assign the label k if $\mu_v(k) \geq \alpha$. The procedure terminates when all nodes have been assigned a label. Our analysis closely follows that of Kleinberg and Tardos [1999].

Lemma A.2.1 *The probability that a node i is assigned label k by the randomized procedure is $\mu_v(k)$.*

Proof The probability that an unassigned node is assigned label k during one phase is $\frac{1}{K}\mu_v(k)$, which is proportional to $\mu_v(k)$. By symmetry, the probability that a node is assigned label k over all phases is exactly $\mu_v(k)$. ∎

Lemma A.2.2 *The probability that all nodes in a clique c are assigned label k by the procedure is at least $\frac{1}{|c|}\mu_c(k)$.*

Proof For a single phase, the probability that all nodes in a clique c are assigned label k if none of the nodes were previously assigned is $\frac{1}{K}\min_{i \in c} \mu_v(k) = \frac{1}{K}\mu_c(k)$. The probability that *at least one* of the nodes will be assigned label k in a phase is $\frac{1}{K}(\max_{i \in c} \mu_v(k))$. The probability that *none* of the nodes in the clique will be assigned *any* label in one phase is $1 - \frac{1}{K}\sum_{k=1}^{K} \max_{i \in c} \mu_v(k)$.

Nodes in the clique c will be assigned label k by the procedure if they are assigned label k in one phase. (They can also be assigned label k as a result of several phases, but we can ignore this possibility for the purposes of the lower bound.) The probability that all the

nodes in c will be assigned label k by the procedure in a single phase is:

$$\sum_{j=1}^{\infty} \frac{1}{K}\mu_c(k)\left(1 - \frac{1}{K}\sum_{k=1}^{K}\max_{i\in c}\mu_v(k)\right)^{j-1} = \frac{\mu_c(k)}{\sum_{k=1}^{K}\max_{i\in c}\mu_v(k)}$$
$$\geq \frac{\mu_c(k)}{\sum_{k=1}^{K}\sum_{i\in c}\mu_v(k)} = \frac{\mu_c(k)}{\sum_{i\in c}\sum_{k=1}^{K}\mu_v(k)} = \frac{\mu_c(k)}{|c|}.$$

Above, we first used the fact that for $d < 1$, $\sum_{i=0}^{\infty} d^i = \frac{1}{1-d}$, and then upper-bounded the $\max$ of the set of positive $\mu_v(k)$'s by their sum. ∎

Theorem A.2.3 *The expected cost of the assignment found by the randomized procedure given a solution μ to the linear program in Eq. (A.5) is at least $\sum_{v\in\mathcal{V}}\sum_{k=1}^{K} g_v(k)\mu_v(k) + \sum_{c\in\mathcal{C}\setminus\mathcal{V}}\frac{1}{|c|}\sum_{k=1}^{K} g_c(k)\mu_c^k$.*

Proof This is immediate from the previous two lemmas.

The only difference between the expected cost of the rounded solution and the (non-integer) optimal solution is the $\frac{1}{|c|}$ factor in the second term. By picking $m = \max_{c\in\mathcal{C}}|c|$, we have that the rounded solution is at most m times worse than the optimal solution produced by the LP of Eq. (A.5). ∎

We can also derandomize this procedure to get a deterministic algorithm with the same guarantees, using the method of conditional probabilities, similar in spirit to the approach of Kleinberg and Tardos [1999].

Note that the approximation factor of m applies, in fact, only to the clique potentials. Thus, if we compare the log-probability of the optimal MAP solution and the log-probability of the assignment produced by this randomized rounding procedure, the terms corresponding to the log-partition-function and the node potentials are identical. We obtain an additive error (in log-probability space) only for the clique potentials. As node potentials are often larger in magnitude than clique potentials, the fact that we incur no loss proportional to node potentials is likely to lead to smaller errors in practice. Along similar lines, we note that the constant factor approximation is smaller for smaller cliques; again, we observe, the potentials associated with large cliques are typically smaller in magnitude, reducing further the actual error in practice.

A.2.3 Derivation of the factored primal and dual max-margin QP

Using Assumptions 7.4.1 and 7.4.4, we have the dual of the LP used to represent the interior max subproblem $\max_{\mathbf{y}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$ in Eq. (3.2):

$$\begin{aligned}
\min \quad & \sum_{v \in \mathcal{V}} \xi_{i,v} \\
\text{s.t.} \quad & -\mathbf{w}^\top \mathbf{f}_{i,v}(k) - \sum_{c \supset v} m_{i,c,v}(k) \geq \ell_{i,v}(k) - \xi_{i,v}, \quad \forall i, v \in \mathcal{V}^{(i)}, k; \\
& -\ddot{\mathbf{w}}^\top \ddot{\mathbf{f}}_{i,c}(k) + \sum_{v \in c} m_{i,c,v}(k) \geq \ell_{i,c}(k), \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, k; \\
& m_{i,c,v}(k) \geq 0, \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, v \in c, k;
\end{aligned} \tag{A.6}$$

where $\mathbf{f}_{i,c}(k) = \mathbf{f}_{i,c}(k, \ldots, k)$ and $\ell_{i,c}(k) = \ell_{i,c}(k, \ldots, k)$. In the dual, we have a variable $\xi_{i,v}$ for each normalization constraint in Eq. (7.1) and variables $m_{i,c,v}(k)$ for each of the inequality constraints.

Substituting this dual into Eq. (5.1), we obtain:

$$\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_i \xi_i \\
\text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq \sum_{v \in \mathcal{V}^{(i)}} \xi_{i,v}, \quad \forall i; \\
& -\mathbf{w}^\top \mathbf{f}_{i,v}(k) - \sum_{c \supset v} m_{i,c,v}(k) \geq \ell_{i,v}(k) - \xi_{i,v}, \quad \forall i, v \in \mathcal{V}^{(i)}, k; \\
& -\ddot{\mathbf{w}}^\top \ddot{\mathbf{f}}_{i,c}(k) + \sum_{v \in c} m_{i,c,v}(k) \geq \ell_{i,c}(k), \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, k; \\
& m_{i,c,v}(k) \geq 0, \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, v \in c, k; \\
& \ddot{\mathbf{w}} \geq 0.
\end{aligned} \tag{A.7}$$

Now let $\xi_{i,v} = \xi'_{i,v} + \mathbf{w}^\top \mathbf{f}_{i,v}(\mathbf{y}_v^{(i)}) + \sum_{c \supset v} \ddot{\mathbf{w}}^\top \ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)})/|c|$ and $m_{i,c,v}(k) = m'_{i,c,v}(k) + \ddot{\mathbf{w}}^\top \ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)})/|c|$. Re-expressing the above QP in terms of these new variables, we get:

$$
\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\
\text{s.t.} \quad & \xi_i \geq \sum_{v\in\mathcal{V}^{(i)}} \xi'_{i,v}, \quad \forall i; \\
& \mathbf{w}^\top \Delta\mathbf{f}_{i,v}(k) - \sum_{c\supset v} m'_{i,c,v}(k) \geq \ell_{i,v}(k) - \xi'_{i,v}, \quad \forall i, v \in \mathcal{V}^{(i)}, k; \\
& \ddot{\mathbf{w}}^\top \Delta\ddot{\mathbf{f}}_{i,c}(k) + \sum_{v\in c} m'_{i,c,v}(k) \geq \ell_{i,c}(k), \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, k; \\
& m'_{i,c,v}(k) \geq -\ddot{\mathbf{w}}^\top \ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)})/|c|, \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, v \in c, k; \\
& \ddot{\mathbf{w}} \geq 0.
\end{aligned}
\tag{A.8}
$$

Since $\xi_i = \sum_{i,v\in\mathcal{V}^{(i)}} \xi'_{i,v}$ at the optimum, we can eliminate ξ_i and the corresponding set of constraints to get the formulation in Eq. (7.4), repeated here for reference:

$$
\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i,v\in\mathcal{V}^{(i)}} \xi_{i,v} \\
\text{s.t.} \quad & \mathbf{w}^\top \Delta\mathbf{f}_{i,v}(k) - \sum_{c\supset v} m_{i,c,v}(k) \geq \ell_{i,v}(k) - \xi_{i,v}, \quad \forall i, v \in \mathcal{V}^{(i)}, k; \\
& \ddot{\mathbf{w}}^\top \Delta\ddot{\mathbf{f}}_{i,c}(k) + \sum_{v\in c} m_{i,c,v}(k) \geq \ell_{i,c}(k), \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, k; \\
& m_{i,c,v}(k) \geq -\ddot{\mathbf{w}}^\top \ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)})/|c|, \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, v \in c, k; \\
& \ddot{\mathbf{w}} \geq 0.
\end{aligned}
\tag{A.9}
$$

Now the dual of Eq. (A.9) is given by:

$$\max \quad \sum_{i,c\in\mathcal{C}^{(i)},\,k} \mu_{i,c}(k)\ell_{i,c}(k) - \frac{1}{2}\left|\left|\sum_{i,v\in\mathcal{V}^{(i)},\,k} \mu_{i,v}(k)\Delta\dot{\mathbf{f}}_{i,v}\right|\right|^2 \tag{A.10}$$

$$-\frac{1}{2}\left|\left|\ddot{\tau} + \sum_{i,c\in\mathcal{C}^{(i)}\setminus\mathcal{V}^{(i)},v\in c,\,k} \lambda_{i,c,v}(k)\ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)})/|c| + \sum_{i,c\in\mathcal{C}^{(i)},\,k} \mu_{i,c}(k)\Delta\ddot{\mathbf{f}}_{i,c}(k)\right|\right|^2$$

$$\text{s.t.} \quad \mu_{i,c}(k) \geq 0, \quad \forall i,\ \forall c \in \mathcal{C}^{(i)},\ k; \qquad \sum_{k=1}^{K} \mu_{i,v}(k) = C, \quad \forall i,\ \forall v \in \mathcal{V}^{(i)};$$

$$\mu_{i,c}(k) - \mu_{i,v}(k) = \lambda_{i,c,v}(k), \quad \forall i,\ \forall c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)},\ v \in c,\ k;$$

$$\lambda_{i,c,v}(k) \geq 0 \quad \forall i,\ \forall c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)},\ v \in c,\ k,$$

$$\ddot{\tau} \geq 0.$$

In this dual, μ correspond to the first two sets of constraints, while λ and $\ddot{\tau}$ correspond to third and fourth set of constraints. Using the substitution

$$\ddot{\nu} = \ddot{\tau} + \sum_{i,c\in\mathcal{C}^{(i)}\setminus\mathcal{V}^{(i)},v\in c,\,k} \lambda_{i,c,v}(k)\ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)})/|c|$$

and the fact that $\lambda_{i,c,v}(k) \geq 0$ and $\ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)}) \geq 0$, we can eliminate λ and $\ddot{\tau}$, as well as divide μ's by C, and re-express the above QP as:

$$\max \quad \sum_{i,c\in\mathcal{C}^{(i)},\,k} \mu_{i,c}(k)\ell_{i,c}(k) - \frac{1}{2}C\left|\left|\sum_{i,v\in\mathcal{V}^{(i)},\,k} \mu_{i,v}(k)\Delta\dot{\mathbf{f}}_{i,v}\right|\right|^2 - \frac{1}{2}C\left|\left|\ddot{\nu} + \sum_{i,c\in\mathcal{C}^{(i)},\,k} \mu_{i,c}(k)\Delta\ddot{\mathbf{f}}_{i,c}(k)\right|\right|^2$$

$$\text{s.t.} \quad \mu_{i,c}(k) \geq 0, \quad \forall i,\ \forall c \in \mathcal{C}^{(i)},\ k; \qquad \sum_{k=1}^{K} \mu_{i,v}(k) = 1, \quad \forall i,\ \forall v \in \mathcal{V}^{(i)};$$

$$\mu_{i,c}(k) \leq \mu_{i,v}(k), \quad \forall i,\ \forall c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)},\ v \in c,\ k; \qquad \ddot{\nu} \geq 0.$$

Bibliography

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. *Proc. ICML*.

Ahuja, R., & Orlin, J. (2001). Inverse optimization, Part I: Linear programming and general problem. *Operations Research*, *35*, 771–783.

Altschul, S., Madden, T., Schaffer, A., Zhang, A., Miller, W., & Lipman (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleid Acids Res.*, *25*, 3389–3402.

Altun, Y., Smola, A., & Hofmann, T. (2004). Exponential families for conditional random fields. *Proc. UAI*.

Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003). Hidden markov support vector machines. *Proc. ICML*.

Bach, F., & Jordan, M. (2001). Thin junction trees. *NIPS*.

Bach, F., & Jordan, M. (2003). Learning spectral clustering. *NIPS*.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. Addison-Wesley-Longman.

Bairoch, A., & Apweiler, R. (2000). The Swiss-Prot protein sequence database and its supplement trembl. *Nucleic Acids Res.*, *28*, 45–48.

Baldi, P., Cheng, J., & Vullo, A. (2004). Large-scale prediction of disulphide bond connectivity. *Proc. NIPS*.

Baldi, P., & Pollastri, G. (2003). The principled design of large-scale recursive neural network architectures dag-rnns and the protein structure prediction problem. *Journal of Machine Learning Research*, *4*, 575–602.

Bansal, N., Blum, A., & Chawla, S. (2002). Correlation clustering. *FOCS*.

Bar-Hillel, A., Hertz, T., Shental, N., & Weinshall, D. (2003). Learning distance functions using equivalence relations. *Proc. ICML*.

Bartlett, P., Collins, M., Taskar, B., & McAllester, D. (2004). Exponentiated gradient algorithms for large-margin structured classification. *NIPS*.

Bartlett, P. L., Jordan, M. I., & McAuliffe, J. D. (2003). *Convexity, classification, and risk bounds* (Technical Report 638). Department of Statistics, U.C. Berkeley.

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, *24*.

Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shindyalov, I., & Bourne, P. (2000). The protein data bank.

Bertsekas, D. (1999). *Nonlinear programming*. Belmont, MA: Athena Scientific.

Bertsimas, D., & Tsitsiklis, J. (1997). *Introduction to linear programming*. Athena Scientific.

Besag, J. E. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, *48*.

Bishop, C. (1995). *Neural networks for pattern recognition*. Oxford, UK: Oxford University Press.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *Proc. COLT*.

Bouman, C., & Shapiro, M. (1994). A multiscale random field model for bayesian image segmentation. *IP*, *3*(2).

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Boykov, Y., & Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in computer vision. *PAMI*.

Boykov, Y., Veksler, O., & Zabih, R. (1999a). Fast approximate energy minimization via graph cuts. *ICCV*.

Boykov, Y., Veksler, O., & Zabih, R. (1999b). Markov random fields with efficient approximations. *CVPR*.

Burton, D., & Toint, P. L. (1992). On an instance of the inverse shortest paths problem. *Mathematical Programming*, *53*, 45–61.

Ceroni, A., Frasconi, P., Passerini, A., & Vullo, A. (2003). Predicting the disulfide bonding state of cysteines with combinations of kernel machines. *Journal of VLSI Signal Processing*, *35*, 287–295.

Chakrabarti, S., Dom, B., & Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. *SIGMOD*.

Chapelle, O., Weston, J., & Schoelkopf, B. (2002). Cluster kernels for semi-supervised learning. *Proc. NIPS*.

Charikar, M., Guruswami, V., & Wirth, A. (2003). Clustering with qualitative information. *FOCS*.

Charniak, E. (1993). *Statistical language learning*. MIT Press.

Clark, S., & Curran, J. R. (2004). Parsing the wsj using ccg and log-linear models. *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL '04)*.

Collins, M. (1999). *Head-driven statistical models for natural language parsing*. Doctoral dissertation, University of Pennsylvania.

Collins, M. (2000). Discriminative reranking for natural language parsing. *ICML 17* (pp. 175–182).

Collins, M. (2001). Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. *IWPT*.

Collins, M. (2004). Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In H. Bunt, J. Carroll and G. Satta (Eds.), *New developments in parsing technology*. Kluwer.

Corduneanu, A., & Jaakkola, T. (2003). On information regularization. *Proc. UAI*.

Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2001). *Introduction to algorithms*. MIT Press. 2nd edition.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. New York: Wiley.

Cowell, R., Dawid, A., Lauritzen, S., & Spiegelhalter, D. (1999). *Probabilistic networks and expert systems*. New York: Springer.

Crammer, K., & Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, *2*(5), 265–292.

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., & Slattery, S. (1998). Learning to extract symbolic knowledge from the world wide web. *Proc AAAI98* (pp. 509–516).

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press.

DeGroot, M. H. (1970). *Optimal statistical decisions*. New York: McGraw-Hill.

Della Pietra, S., Della Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, *19*(4), 380–393.

Demaine, E. D., & Immorlica, N. (2003). Correlation clustering with partial information. *APPROX*.

Dempster, P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, *39*.

Devroye, L., Györfi, L., & Lugosi, G. (1996). *Probabilistic theory of pattern recognition*. New York: Springer-Verlag.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification*. New York: Wiley Interscience. 2nd edition.

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). *Biological sequence analysis*. Cambridge University Press.

Edmonds, J. (1965). Maximum matching and a polyhedron with 0-1 vertices. *Journal of Research at the National Bureau of Standards*, *69B*, 125–130.

Egghe, L., & Rousseau, R. (1990). *Introduction to informetrics*. Elsevier.

Emanuel, D., & Fiat, A. (2003). Correlation clustering - minimizing disagreements on arbitrary weighted graphs. *ESA*.

Fariselli, P., & Casadio, R. (2001). Prediction of disulfide connectivity in proteins. *Bioinformatics*, *17*, 957–964.

Fariselli, P., Martelli, P., & Casadio, R. (2002). A neural network-based method for predicting the disulfide connectivity in proteins. *Knowledge based intelligent information engineering systems and allied technologies (KES 2002)*, *1*, 464–468.

Fariselli, P., Ricobelli, P., & Casadio, R. (1999). Role of evolutionary information in predicting the disulfide-bonding state of cysteine in proteins. *Proteins*, *36*, 340–346.

Fiser, A., & Simon, I. (2000). Predicting the oxidation state of cysteines by multiple sequence alignment. *Bionformatics*, *3*, 251–256.

Forsyth, D. A., & Ponce, J. (2002). *Computer vision: A modern approach*. Prentice Hall.

Frasconi, P., Gori, M., & Sperduti, A. (1998). A general framework for adaptive structures. *IEEE Transactions on Neural Networks*.

Frasconi, P., Passerini, A., & Vullo, A. (2002). A two stage svm architecture for predicting the disulfide bonding state of cysteines. *Proceedings of IEEE Neural Network for signal processing conference*.

Freund, Y., & Schapire, R. (1998). Large margin classification using the perceptron algorithm. *Computational Learing Theory*.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. *Proc. IJCAI99* (pp. 1300–1309). Stockholm, Sweden.

Friess, T., Cristianini, N., & Campbell, C. (1998). The kernel adatron algorithm: a fast and simple learning procedure for support vector machine. *Proc. ICML*.

Gabow, H. (1973). *Implementation of algorithms for maximum matching on nonbipartite graphs*. Doctoral dissertation, Stanford University.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability*. Freeman.

Geman, S., & Johnson, M. (2002). Dynamic programming for parsing and estimation of stochastic unification-based grammars. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Getoor, L., Friedman, N., Koller, D., & Taskar, B. (2002). Learning probabilistic models of link structure. *Journal of Machine Learning Research, 8*.

Getoor, L., Segal, E., Taskar, B., & Koller, D. (2001). Probabilistic models of text and link structure for hypertext classification. *Proc. IJCAI01 Workshop on Text Learning: Beyond Supervision*. Seattle, Wash.

Greig, D. M., Porteous, B. T., & Seheult, A. H. (1989). Exact maximum a posteriori estimation for binar images. *J. R. Statist. Soc. B, 51*.

Guestrin, C., Koller, D., Parr, R., & Venkataraman, S. (2003). Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research, 19*.

Gusfield, D. (1997). *Algorithms on strings, trees, and sequences : Computer science and computational biology*. Cambridge University Press.

Harrison, P., & Sternberg, M. (1994). Analysis and classification of disulphide connectivity in proteins. the entropic effect of cross-linkage. *Journal of Molecular Biology*, *244*.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. New York: Springer-Verlag.

Haussler, D. (1999). *Convolution kernels on discrete structures* (Technical Report). UC Santa Cruz.

Heuberger, C. (2004). Inverse combinatorial optimization: A survey on problems, methods, and results. *Journal of Combinatorial Optimization*, *8*.

Hochbaum, D. S. (Ed.). (1997). *Approximation algorithms for NP-hard problems*. PWS Publishing Company.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *Proc. ICML99* (pp. 200–209). Morgan Kaufmann Publishers, San Francisco, US.

Johnson, M. (2001). Joint and conditional estimation of tagging and parsing models. *ACL 39*.

Johnson, M., Geman, S., Canon, S., Chi, Z., & Riezler, S. (1999). Estimators for stochastic "unification-based" grammars. *Proceedings of ACL 1999*.

Kabsch, W., & Sander, C. (1983). Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features.

Kaplan, R., Riezler, S., King, T., Maxwell, J., Vasserman, A., & Crouch, R. (2004). Speed and accuracy in shallow and deep stochastic parsing. *Proceedings of HLT-NAACL'04*.

Kassel, R. (1995). *A comparison of approaches to on-line handwritten character recognition*. Doctoral dissertation, MIT Spoken Language Systems Group.

Kivinen, J., & Warmuth, M. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, *132*(1), 1–63.

Klein, D., & Manning, C. D. (2003). Accurate unlexicalized parsing. *ACL 41* (pp. 423–430).

Kleinberg, J., & Tardos, E. (1999). Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. *FOCS*.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, *46*(5), 604–632.

Klepeis, J., & Floudas, C. (2003). Prediction of β-sheet topology and disulfide bridges in polypeptides. *Journal of Computational Chemistry*, *24*, 191–208.

Koller, D., & Pfeffer, A. (1998). Probabilistic frame-based systems. *Proc. AAAI98* (pp. 580–587). Madison, Wisc.

Kolmogorov, V., & Zabih, R. (2002). What energy functions can be minimized using graph cuts? *PAMI*.

Kumar, S., & Hebert, M. (2003). Discriminative fields for modeling spatial dependencies in natural images. *NIPS*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*.

Lafferty, J., Zhu, X., & Liu, Y. (2004). Kernel conditional random fields: Representation and clique selection. *ICML*.

Lawler, E. (1976). *Combinatorial optimization: Networks and matroids*. New York: Holt, Rinehart and Winston.

Leslie, C., Eskin, E., & Noble, W. (2002). The spectrum kernel: a string kernel for svm protein classification. *Proc. Pacific Symposium on Biocomputing*.

Liu, Z., & Zhang, J. (2003). On inverse problems of optimum perfect matching. *Journal of Combinatorial Optimization*, *7*.

Lodhi, H., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2000). Text classification using string kernels. *NIPS*.

Luenberger, D. (1997). *Investment science*. Oxford University Press.

Manning, C., & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge, Massachusetts: The MIT Press.

Marcus, M. P., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, *19*(2), 313–330.

Markowitz, H. (1991). *Portfolio selection: Efficient diversification of investments*. Basil Blackwell.

Martelli, P., Fariselli, P., Malaguti, L., & Casadio, R. (2002). Prediction of the disulfide-bonding state of cysteines in proteins at 88% accuracy. *Protein Science*, *11*, 27359.

Martin, D., Fowlkes, C., Tal, D., & Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *Proc. ICCV*.

Matsumura, M., Signor, G., & Matthews, B. (1989). Substantial increase of protein stability by multiple disulfide bonds. *Nature*, *342*, 291:293.

Matusov, E., Zens, R., & Ney, H. (2004). Symmetric word alignments for statistical machine translation. *Proc. COLING*.

McCallum, A. (2003). Efficiently inducing features of conditional random fields. *Proc. UAI*.

McCallum, A., & Wellner, B. (2003). Toward conditional models of identity uncertainty with application to proper noun coreference. *IJCAI Workshop on Information Integration on the Web*.

Mitchell, T. (1997). *Machine learning*. McGraw-Hill.

Miyao, Y., & Tsujii, J. (2002). Maximum entropy estimation for feature forests. *Proceedings of Human Language Technology Conference (HLT 2002)*.

Murphy, K. P., Weiss, Y., & Jordan, M. I. (1999). Loopy belief propagation for approximate inference: an empirical study. *Proc. UAI99* (pp. 467–475).

Needleman, S., & Wunsch, C. (1970). A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology, 48*.

Nemhauser, G. L., & Wolsey, L. A. (1999). *Integer and combinatorial optimization*. New York: J. Wiley.

Neville, J., & Jensen, D. (2000). Iterative classification in relational data. *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data* (pp. 13–20).

Ng, A., & Jordan, M. (2001). On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. *NIPS*.

Ng, A. Y., & Russell, S. (2000). Algorithms for inverse reinforcement learning. *Proc. ICML*.

Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine Learning, 39*.

Nocedal, J., & Wright, S. J. (1999). *Numerical optimization*. New York: Springer.

Papadimitriou, C., & Steiglitz, K. (1982). *Combinatorial optimization: Algorithms and complexity*. Englewood Cliffs, NJ: Prentice-Hall.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems*. San Francisco: Morgan Kaufmann.

Pinto, D., McCallum, A., Wei, X., & Croft, W. B. (2003). Table extraction using conditional random fields. *Proc. ACM SIGIR*.

Platt, J. (1999). Using sparseness and analytic QP to speed training of support vector machines. *NIPS*.

Potts, R. B. (1952). Some generalized order-disorder transformations. *Proc. Cambridge Phil. Soc.*, *48*.

Quinlan, J. R. (2001). Induction of decision trees. *Machine Learning*, *1*, 81–106.

Schrijver, A. (2003). *Combinatorial optimization: Polyhedra and efficiency*. Springer.

Sha, F., & Pereira, F. (2003). Shallow parsing with conditional random fields. *Proc. HLT-NAACL*.

Shawe-Taylor, J., Bartlett, P. L., Williamson, R. C., & Anthony, M. (1998). Structural risk minimization over data-dependent hierarchies. *IEEE Trans. on Information Theory*, *44*(5), 1926–1940.

Shen, L., Sarkar, A., & Joshi, A. K. (2003). Using ltag based features in parse reranking. *Proc. EMNLP*.

Slattery, S., & Mitchell, T. (2000). Discovering test set regularities in relational domains. *Proc. ICML00* (pp. 895–902).

Smith, T., & Waterman, M. (1981). Identification of common molecular subsequences. *J. Mol. Biol.*, *147*, 195–197.

Sutton, C., Rohanimanesh, K., & McCallum, A. (2004). Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Proc. ICML*.

Szummer, M., & Jaakkola, T. (2001). Partially labeled classification with Markov random walks. *Proc. NIPS*.

Tarantola, A. (1987). *Inverse problem theory: Methods for data fitting and model parameter estimation*. Amsterdam: Elsevier.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. *UAI*.

Taskar, B., Chatalbashev, V., & Koller, D. (2004a). Learning associative Markov networks. *Proc. ICML*.

Taskar, B., Guestrin, C., & Koller, D. (2003a). Max margin Markov networks. *NIPS*.

Taskar, B., Klein, D., Collins, M., Koller, D., & Manning, C. (2004b). Max margin parsing. *EMNLP*.

Taskar, B., Segal, E., & Koller, D. (2001). Probabilistic classification and clustering in relational data. *Proc. IJCAI01* (pp. 870–876). Seattle, Wash.

Taskar, B., Wong, M., Abbeel, P., & Koller, D. (2003b). Link prediction in relational data. *Proc. NIPS*.

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. *NAACL 3* (pp. 252–259).

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. *Twenty-first international conference on Machine learning*.

Valiant, L. G. (1979). The complexity of computing the permanent. *Theoretical Computer Science*, *8*, 189–201.

Vapnik, V. (1995). *The nature of statistical learning theory*. New York, New York: Springer-Verlag.

Varma, A., & Palsson, B. (1994). Metabolic flux balancing: Basic concepts, scientific and practical use. *Bio/Technology*, *12*.

Vazquez, A., Flammini, A., Maritan, A., & Vespignani, A. (2003). Global protein function prediction from protein-protein interaction networks$h$. *Nature Biotechnology*, *6*.

Veksler, O. (1999). *Efficient graph-based energy minimization methods in computer vision*. Doctoral dissertation, Cornell University.

Vullo, A., & Frasconi, P. (2004). Disulfide connectivity prediction using recursive neural networks and evolutionary information. *Bioinformatics*, *20*, 653–659.

Wahba, G., Gu, C., Wang, Y., & Chappell, R. (1993). Soft classification, a.k.a. risk estimation, via penalized log likelihood and smoothing spline analysis of variance. *Computational Learning Theory and Natural Learning Systems*.

Wainwright, M., Jaakkola, T., & Willsky, A. (2002). Map estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. *Proc. Allerton Conference on Communication, Control and Computing*.

Wainwright, M., & Jordan, M. I. (2003). Variational inference in graphical models: The view from the marginal polytope. *Proc. Allerton Conference on Communication, Control and Computing*.

Weston, J., & Watkins, C. (1998). *Multi-class support vector machines* (Technical Report CSD-TR-98-04). Department of Computer Science, Royal Holloway, University of London.

Xing, E., Ng, A., Jordan, M., & Russell, S. (2002). Distance metric learning, with application to clustering with side-information. *NIPS*.

Yang, Y., Slattery, S., & Ghani, R. (2002). A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, *18*(2).

Yedidia, J., Freeman, W., & Weiss, Y. (2000). Generalized belief propagation. *NIPS*.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, *10*, 189–208.

Zhang, J., & Ma, Z. (1996). A network flow method for solving inverse combinatorial optimization problems. *Optimization*, *37*, 59–72.

Zhang, T. (2002). Covering number bounds of certain regularized linear function classes. *Journal of Machine Learning Research*, *2*, 527–550.

Zhu, J., & Hastie, T. (2001). Kernel logistic regression and the import vector machine. *Proc. NIPS*.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. *Proc. ICML*.