

Discriminative Machine Learning with Structure

by

Simon Lacoste-Julien

B.Sc.H. (McGill University) 2003

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

Doctor of Philosophy

in

COMPUTER SCIENCE
and the Designated Emphasis in
Communication, Computation and Statistics

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Michael I. Jordan, Chair

Professor Peter L. Bartlett

Professor Peter J. Bickel

Professor Dan Klein

Fall 2009

The dissertation of Simon Lacoste-Julien is approved:

Professor Michael I. Jordan, Chair Date

Professor Peter L. Bartlett Date

Professor Peter J. Bickel Date

Professor Dan Klein Date

University of California, Berkeley

Fall 2009

Discriminative Machine Learning with Structure

Copyright © 2009

by

Simon Lacoste-Julien

Abstract

Discriminative Machine Learning with Structure

by

Simon Lacoste-Julien

Doctor of Philosophy in Computer Science

and the Designated Emphasis in Communication, Computation and Statistics

University of California, Berkeley

Professor Michael I. Jordan, Chair

Some of the best performing classifiers in modern machine learning have been designed using discriminative learning, as exemplified by Support Vector Machines. The ability of discriminative learning to use flexible features via the kernel trick has enlarged the possible set of applications for machine learning. With the expanded range of possible applications though, it has become apparent that real world data exhibits more structure than has been assumed by classical methods. In this thesis, we show how to extend the discriminative learning framework to exploit different types of structure: on one hand, the structure on outputs, such as the combinatorial structure in word alignment; on the other hand, a latent variable structure on inputs, such as in text document classification.

In the context of structured output classification, we present a scalable algorithm for maximum-margin estimation of structured output models, including an important class of Markov networks and combinatorial models. We formulate the estimation problem as a convex-concave saddle-point problem that allows us to use simple projection methods based on the dual extragradient algorithm of Nesterov. We analyze the convergence of the method and present experiments on two very different struc-

tured prediction tasks: 3D image segmentation and word alignment. We then show how one can obtain state-of-the-art results for the word alignment task by formulating it as a quadratic assignment problem within our discriminative learning framework.

In the context of latent variable models, we present DiscLDA, a discriminative variant of the Latent Dirichlet Allocation (LDA) model which has been popular to model collections of text documents or images. In DiscLDA, we introduce a class-dependent linear transformation on the topic mixture proportions of LDA and estimate it discriminatively by maximizing the conditional likelihood. By using the transformed topic mixture proportions as a new representation of documents, we obtain a supervised dimensionality reduction algorithm that uncovers the latent structure in a document collection while preserving predictive power for the task of classification. Our experiments on the 20 Newsgroups document classification task show how our model can identify shared topics across classes as well as discriminative class-dependent topics.

Professor Michael I. Jordan, Chair

Date

Acknowledgements

I am profoundly grateful to my advisor, Michael Jordan, for providing a stimulating research environment with plenty of opportunities, for his continuous support and his insightful guidance in research as well as in life. Little did I know six years ago when I received his phone call in (fluent) French in the middle of a dinner with my parents that I would embark upon such a journey in the world of machine learning. I will also always remember the ineluctable feeling of motivation and renewed energy resulting from an advising meeting with him. Mike's teaching and research excellence have been a tremendous source of inspiration.

I would like to thank the members of my thesis committee, Peter Bartlett, Peter Bickel and Dan Klein. Peter (Bartlett) launched my interest for structured prediction problems during a few brainstorming sessions for my first machine learning class project. I have learned a lot from Dan's incredible practical insights on machine learning and natural language processing. He has been a driving force for a significant part of the work in this thesis, and I am also grateful for the experience of being a teaching assistant for him – Dan is a truly gifted and inspiring teacher. Finally, I thank Peter (Bickel) for sharing his statistical insights with me.

I was quite fortunate to have the opportunity to work with Ben Taskar when he visited Berkeley as a postdoc. Our research interests were wonderfully aligned and he basically became a second academic mentor for me. Two chapters of this thesis are based on work done in collaboration with Ben, and I am eternally grateful for everything he taught me during this time. Bol'shoe spasibo, Ben, for being such a wonderful friend and mentor, and for making me discover Radio Paradise during those late nights in the Berkeley lab.

I have had the chance to interact with many talented colleagues at UC Berkeley

and in the wider academic community, and I would like to recognize their contribution to my research. A special thanks to Fei Sha for helping me turn a vague research agenda into a concrete project which resulted in the DiscLDA model. I have enjoyed countless discussions with colleagues and friends Guillaume Obozinski and Romain Thibaux, who have accompanied me since the beginning of my PhD. I have also benefited from the research interactions with Deepak Agarwal, Francis Bach, Yoshua Bengio, David Blei, Wray Buntine, Vanja Josifovski, Percy Liang, Iain Murray, Ruslan Salakhutdinov, Erik Sudderth, Yee Whye Teh, Andrew Tomkins, Martin Wainwright and the other members of the SAIL research group.

I am grateful to have been exposed early to the wonders of research while at McGill University. I thank Prakash Panangaden for introducing me to my first research project, for his inspiring teaching, and for being instrumental in making me discover the world outside of Québec. I thank Hans Vangheluwe for being a wonderful undergraduate mentor and for his contagious enthusiasm for research. Finally, I thank Doina Precup for getting me excited about Artificial Intelligence and having suggested a very good graduate supervisor.

A portion of this thesis was written at the University of Cambridge while I was being supported by Zoubin Ghahramani. I would like to thank him for his understanding, his support and for providing such a wonderful environment. Zoubin is also a truly inspiring research advisor. Another special thanks to Percy Liang for dealing with the paperwork while I was overseas.

I would like to acknowledge the financial support that I have received throughout my graduate studies, from fellowships from the National Sciences and Engineering Research Council of Canada and from the Fonds québécois de la recherche sur la nature et les technologies, and from a grant from the National Science Foundation.

Thanks to my new friends from Berkeley who helped me to stay sane during difficult times and enjoy a life outside of the Ivory Tower: Neil and Andrea, Morgan,

Sasha and many others. Sasha, I hope that you can find your path. Dasha, spasibo for everything that we have shared during the major part of my PhD and that we have learned together. Merci to my old friends from Montréal with whom I have succeeded to stay connected despite the distance, and to my dear family.

Dedicated to my dear friend and mentor, Ben.

Contents

List of Figures	ix
List of Tables	xi
1 Introduction	1
2 Scalable Algorithm for Discriminative Structured Prediction	4
2.1 Background	4
2.2 Structured output models	8
2.2.1 Tree-structured Markov networks	8
2.2.2 Markov networks with submodular potentials	10
2.2.3 Matchings	12
2.2.4 General structure	13
2.3 Large margin estimation	14
2.4 Saddle-point problems and the dual extragradient method	19
2.4.1 Dual extragradient	21
2.4.1.1 Convergence	22
2.4.1.2 Projections	24
2.4.2 Non-Euclidean dual extragradient	26
2.5 Memory-efficient formulation	31

2.5.1	Kernels	32
2.6	Experiments	34
2.6.1	Object segmentation	35
2.6.2	Word alignment	38
2.7	Related work and discussion	42
2.7.1	Online methods	43
2.8	Summary	46
	Appendix 2.A Min-cut polytope projections	47
3	Word Alignment via Quadratic Assignment	50
3.1	Background	50
3.2	Models	53
3.2.1	Fertility	54
3.2.2	First-order interactions	57
3.3	Parameter estimation	59
3.4	Experiments	62
3.4.1	Features and results	63
3.4.1.1	The Basic Matching Model: Edge Features	63
3.4.1.2	The Fertility Model: Node Features	64
3.4.1.3	The First-Order Model: Quadratic Features	65
3.5	Summary	66
4	DiscLDA: Discriminative Dimensionality Reduction for Classification	68
4.1	Background	69
4.2	Model	71
4.2.1	LDA	71
4.2.2	DiscLDA	72

4.3	Inference and learning	76
4.3.1	Approximate inference for learning	78
4.3.2	Approximating the posterior over labels	80
4.3.3	Dimensionality reduction	81
4.4	Experimental results	82
4.4.1	Dataset description	82
4.4.2	Experiments with fixed \mathbf{T}^y	83
4.4.2.1	Text modeling	83
4.4.2.2	Document classification	87
4.4.3	Experiments with discriminatively trained \mathbf{T}^y	89
4.5	Discussion	97
4.5.1	Related work	97
4.5.1.1	Author-topic model	97
4.5.1.2	Supervised topic models	97
4.5.2	Caveats and future directions	98
4.6	Summary	101
Appendix 4.A	Derivations	102
4.A.1	Gradient equations	102
4.A.1.1	Using an EM bound	102
4.A.1.2	Gradient derivation	103
4.A.2	Gibbs sampling equations	106
4.A.3	Dimensionality reduction equation	109
Appendix 4.B	Bridge sampling	109
4.B.1	Application to DiscLDA	114
5	Conclusions	117
5.1	Summary	117

5.2 Future Directions	118
Bibliography	120
Index	129

List of Figures

2.1	Structured prediction applications	11
2.2	Euclidean dual extragradient algorithm	21
2.3	Projection on the matching polytope expressed as a network flow	24
2.4	Non-Euclidean dual extragradient algorithm	26
2.5	Dependency diagram for memory-efficient dual extragradient algorithm	33
2.6	Memory-efficient dual extragradient algorithm	33
2.7	Object segmentation results	37
2.8	Word alignment results	40
3.1	Network flow for word alignment	55
3.2	Word alignment fertility example	56
3.3	Monotonic alignment example	57
3.4	Diamond shape alignment example	62
4.1	LDA graphical model	73
4.2	DiscLDA graphical model	73
4.3	DiscLDA graphical model with auxiliary variable	73
4.4	Simplex transformation interpretation of DiscLDA	74
4.5	t-SNE embedding on 20 Newsgroups of DiscLDA representation	86
4.6	t-SNE embedding on 20 Newsgroups of LDA representation	86

4.7	Binary classification results	91
4.8	t-SNE embedding for binary classification tasks (1)	95
4.9	t-SNE embedding for binary classification tasks (2)	96
4.10	Author-topic graphical model with only one author per document . .	99
4.11	DiscLDA model repeated	99
4.12	Supervised LDA graphical model	99
4.13	Dirichlet-multinomial regression graphical model	99
4.14	Geometric bridge example	113

List of Tables

3.1	AER results on the Hansards task	66
4.1	DiscLDA learning algorithm	78
4.2	20 Newsgroups dataset	83
4.3	Topics learned in the 20 Newsgroups experiment with fixed \mathbf{T}^y matrix	88
4.4	Classification error rates on 20 Newsgroups dataset	89
4.5	Topics learned in the binary classification tasks with \mathbf{T}^y learned dis- criminatively with DiscLDA	93
4.6	Individual topics learned in the binary classification tasks with \mathbf{T}^y learned discriminatively with DiscLDA	94

Chapter 1

Introduction

In supervised learning for classification, the goal is to learn a function from inputs (objects) to discrete outputs (labels), given a training set of already labeled instances. This framework has a vast number of applications, ranging from machine translation to optical character recognition. Two of the major paradigms of machine learning for solving this problem have been the *generative* and the *discriminative* approaches. In the *generative approach* (also called generative learning), one models a *joint* probability distribution on inputs and outputs. The parameters of this distribution are estimated by using a likelihood-based criterion, such as regularized maximum likelihood, or are marginalized out, such as in the Bayesian approach. Prediction is done by computing posterior probabilities based on the probabilistic model. By contrast, in the *discriminative approach*, one directly models the mapping from inputs to outputs, either as a conditional distribution or as a function. The parameters are estimated by optimizing various objectives related to the loss function of the classification task, such as regularized conditional likelihood or a large-margin criterion. Discriminative approaches typically show better classification performance with enough data, as they are better tuned to the prediction task and are more robust to

model misspecification (Ng and Jordan, 2002; Liang and Jordan, 2008). On the other hand, the generative modeling approach provides a natural way to encode more complex structural information about the data such as with the language of probabilistic graphical models (Jordan, 1999). With the widening range of applications considered by machine learning in the last decade, discriminative approaches have been developed to leverage graphical models and be used on prediction tasks for which the labels are structured objects with complex interactions (Lafferty et al., 2001; Taskar et al., 2004b).

The goal of this thesis is to extend the ability of discriminative methods to exploit the structure in the data, akin to generative methods, but maintaining the goal of classification in mind. In this regard, we make the following contributions:

- In Chapter 2, we present a scalable algorithm for maximum-margin estimation of structured output models, including an important class of Markov networks and combinatorial models. We formulate the estimation problem as a convex-concave saddle-point problem that allows us to use simple projection methods based on the dual extragradient algorithm of Nesterov (2003). The projection step can be solved using dynamic programming or combinatorial algorithms for min-cost convex flow, depending on the structure of the problem. We show that this approach provides a memory-efficient alternative to formulations based on reductions to a quadratic program (QP). We analyze the convergence of the method and present experiments on two very different structured prediction tasks: 3D image segmentation and word alignment, illustrating the favorable scaling properties of our algorithm. In particular, we show how we can apply max-margin learning to tens of thousands of sentences for word alignment whereas standard off-the-shelf solvers go out of memory after 1,000 to 2,000 sentences. This work was published in Taskar et al. (2006b).

- In Chapter 3, we explore more in details our learning framework for the word alignment application in natural language processing. We propose a more realistic model which encodes similar linguistic information which was used by previous generative models but was missing in the original discriminative model: fertility and first order interactions. This provides a good example of the power of discriminative learning: with those simple modifications, we can obtain the best alignment error rate published so far on the French-English Hansards dataset. This work was published in [Lacoste-Julien et al. \(2006\)](#).
- In Chapter 4, we turn to the problem of exploiting a latent variable structure on the inputs for the task of classification. We present DiscLDA, a discriminative variant of the Latent Dirichlet Allocation (LDA) model which has been popular for the modeling of collections of text documents or images. We consider the problem of finding a reduced dimensionality representation of text documents which preserves its predictive power for classification. In DiscLDA, we introduce a class-dependent linear transformation on the topic mixture proportions of LDA. We estimate this parameter discriminatively by maximizing the conditional likelihood on the training set. By using the transformed topic mixture proportions as a new representation of documents, we obtain a supervised dimensionality reduction algorithm that uncovers the latent structure in a document collection while preserving predictive power for the task of classification. We compare the predictive power of the latent structure of DiscLDA with unsupervised LDA on the 20 Newsgroups document classification task and show how our model can identify shared topics across classes as well as discriminative class-dependent topics. This work was published in [Lacoste-Julien et al. \(2009\)](#).

Chapter 2

Scalable Algorithm for Discriminative Structured Prediction

2.1 Background

Structured prediction problems are classification or regression problems in which the output variables (the class labels or regression responses) are interdependent. These dependencies may reflect sequential, spatial, recursive or combinatorial structure in the problem domain, and capturing these dependencies is often as important for the purposes of prediction as capturing input-output dependencies. In addition to modeling output correlations, we may wish to incorporate hard constraints between variables. For example, we may seek a model that maps descriptions of pairs of structured objects (shapes, strings, trees, etc.) into alignments of those objects. Real-life examples of such problems include bipartite matchings in alignment of 2D shapes ([Belongie et al., 2002](#)) and word alignment of sentences from a source language to a target language in machine translation ([Matusov et al., 2004](#)) or non-bipartite matchings of residues in disulfide connectivity prediction for proteins ([Baldi et al., 2005](#)). In these

examples, the output variables encode presence of edges in the matching and may obey hard one-to-one matching constraints. The prediction problem in such situations is often solved via efficient combinatorial optimization such as finding the maximum weight matching, where the model provides the appropriate edge weights.

Thus in this thesis we define the term *structured output model* very broadly, as a compact scoring scheme over a (possibly very large) set of combinatorial structures and a method for finding the highest scoring structure. For example, when a probabilistic graphical model is used to capture dependencies in a structured output model, the scoring scheme is specified via a factorized probability distribution for the output variables conditional on the input variables, and the search involves some form of generalized Viterbi algorithm. More broadly, in models based on combinatorial problems, the scoring scheme is usually a simple sum of weights associated with vertices, edges, or other components of a structure; these weights are often represented as parametric functions of the inputs. Given training data consisting of instances labeled by desired structured outputs and a set of features that parameterize the scoring function, the (discriminative) learning problem is to find parameters such that the highest scoring outputs are as close as possible to the desired outputs.

In the case of structured prediction based on graphical models, which encompasses a large proportion of the work to date on structured prediction, two major approaches to discriminative learning have been explored¹: (1) maximum conditional likelihood (Lafferty et al., 2001, 2004) and (2) maximum margin (Collins, 2002; Al-tun et al., 2003; Taskar et al., 2004b). Both approaches are viable computationally for restricted classes of graphical models. In the broader context of the current thesis, however, only the maximum-margin approach appears to be viable for exact inference. In particular, it has been shown that maximum-margin estimation

¹Other approaches not based on graphical models include search-based learning, as in the SEARN algorithm of Daumé III et al. (2009), and energy-based methods (LeCun et al., 2006).

can be formulated as a tractable convex problem — a polynomial-size quadratic program (QP) — in several cases of interest (Taskar et al., 2004a, 2005a); such results are not available for conditional likelihood. Moreover, it is possible to find interesting subfamilies of graphical models for which maximum-margin methods are provably tractable whereas likelihood-based methods are not. For example, for the Markov random fields that arise in object segmentation problems in vision (Kumar and Hebert, 2004; Anguelov et al., 2005) the task of finding the most likely assignment reduces to a min-cut problem. In these prediction tasks, the problem of finding the highest scoring structure is tractable, while computing the partition function is $\#P$ -complete. Essentially, maximum-likelihood estimation requires the partition function, while maximum-margin estimation does not, and thus remains tractable. Polynomial-time sampling algorithms for approximating the partition function for some models do exist (Jerrum and Sinclair, 1993), but have high-degree polynomial complexity and have not yet been shown to be effective for conditional likelihood estimation.

While the reduction to a tractable convex program such as a QP is a significant step forward, it is unfortunately not the case that off-the-shelf QP solvers necessarily provide practical solutions to structured prediction problems. Indeed, despite the reduction to a polynomial number of variables, off-the-shelf QP solvers tend to scale poorly with problem and training sample size for these models. The number of variables is still large and the memory needed to maintain second-order information (for example, the inverse Hessian) is a serious practical bottleneck.

To solve the largest-scale machine learning problems, researchers have often found it expedient to consider simple gradient-based algorithms, in which each individual step is cheap in terms of computation and memory (Platt, 1999; LeCun et al., 1998). Examples of this approach in the structured prediction setting include the Structured Sequential Minimal Optimization algorithm (Taskar et al., 2004b; Taskar, 2004) and

the Structured Exponentiated Gradient algorithm (Bartlett et al., 2005). These algorithms are first-order methods for solving QPs arising from low-treewidth Markov random fields and other decomposable models. In these restricted settings these methods can be used to solve significantly larger problems than can be solved with off-the-shelf QP solvers. These methods are, however, limited in scope in that they rely on dynamic programming to compute essential quantities such as gradients. They do not extend to models where dynamic programming is not applicable, for example, to problems such as matchings and min-cuts. Another line of work in learning structured prediction models aims to approximate the arising QPs via constraint generation (Altun et al., 2003; Tsochantaridis et al., 2005). This approach only requires finding the highest scoring structure in the inner loop and incrementally solving a growing QP as constraints are added.

In this chapter, we present a solution methodology for structured prediction that encompasses a broad range of combinatorial optimization problems, including matchings, min-cuts and other network flow problems. There are two key aspects to our methodology. The first is that we take a novel approach to the formulation of structured prediction problems, formulating them as saddle-point problems. This allows us to exploit recent developments in the optimization literature, where simple gradient-based methods have been developed for solving saddle-point problems (Nesterov, 2003). Moreover, we show that the key computational step in these methods—a certain projection operation—inherits the favorable computational complexity of the underlying optimization problem. This important result makes our approach viable computationally. In particular, for decomposable graphical models, the projection step is solvable via dynamic programming. For matchings and min-cuts, projection involves a min-cost quadratic flow computation, a problem for which efficient, highly-specialized algorithms are available.

The remaining of this chapter is organized as follows. In Section 2.2 we present an

overview of structured prediction, focusing on three classes of tractable optimization problems. Section 2.3 shows how to formulate the maximum-margin estimation problem for these models as a saddle-point problem. In Section 2.4 we discuss the dual extragradient method for solving saddle-point problems and show how it specializes to our setting. We derive a memory-efficient version of the algorithm that requires storage proportional to the number of parameters in the model and is independent of the number of examples in Section 2.5. In Section 2.6 we illustrate the effectiveness of our approach on two very different large-scale structured prediction tasks: 3D image segmentation and word alignment in natural language translation. Finally, Section 2.8 presents our conclusions.

2.2 Structured output models

We begin by discussing three special cases of the general framework that we present subsequently: (1) tree-structured Markov networks, (2) Markov networks with submodular potentials, and (3) a bipartite matching model. Despite significant differences in the formal specification of these models, they share the property that in all cases the problem of finding the highest-scoring output can be formulated as a linear program (LP).

2.2.1 Tree-structured Markov networks

For simplicity of notation, we focus on tree networks, noting in passing that the extension to hypertrees is straightforward. Given N variables, $\mathbf{y} = \{y_1, \dots, y_N\}$, with discrete domains $y_j \in \mathcal{D}_j = \{\alpha_1, \dots, \alpha_{|\mathcal{D}_j|}\}$, we define a joint distribution over

$\mathcal{Y} = \mathcal{D}_1 \times \dots \times \mathcal{D}_N$ via

$$P(\mathbf{y}) \propto \prod_{j \in \mathcal{V}} \phi_j(y_j) \prod_{jk \in \mathcal{E}} \phi_{jk}(y_j, y_k),$$

where $(\mathcal{V} = \{1, \dots, N\}, \mathcal{E} \subset \{jk : j < k, j \in \mathcal{V}, k \in \mathcal{V}\})$ is an undirected graph, and where $\{\phi_j(y_j), j \in \mathcal{V}\}$ are the node potentials and $\{\phi_{jk}(y_j, y_k), jk \in \mathcal{E}\}$ are the edge potentials. We can find the most likely assignment, $\arg \max_{\mathbf{y}} P(\mathbf{y})$, using the Viterbi dynamic programming algorithm for trees. We can also find it using a standard linear programming formulation as follows. We introduce variables $z_{j\alpha}$ to denote indicators $\mathbb{I}(y_j = \alpha)$ for all variables $j \in \mathcal{V}$ and their values $\alpha \in \mathcal{D}_j$. Similarly, we introduce variables $z_{jk\alpha\beta}$ to denote indicators $\mathbb{I}(y_j = \alpha, y_k = \beta)$ for all edges $jk \in \mathcal{E}$ and the values of their nodes, $\alpha \in \mathcal{D}_j, \beta \in \mathcal{D}_k$. We can formulate the problem of finding the maximal probability configuration as follows:

$$\max_{0 \leq z \leq 1} \sum_{j \in \mathcal{V}} \sum_{\alpha \in \mathcal{D}_j} z_{j\alpha} \log \phi_j(\alpha) + \sum_{jk \in \mathcal{E}} \sum_{\alpha \in \mathcal{D}_j, \beta \in \mathcal{D}_k} z_{jk\alpha\beta} \log \phi_{jk}(\alpha, \beta) \quad (2.1)$$

$$\text{s.t.} \quad \sum_{\alpha \in \mathcal{D}_j} z_{j\alpha} = 1, \quad \forall j \in \mathcal{V}; \quad \sum_{\alpha \in \mathcal{D}_j, \beta \in \mathcal{D}_k} z_{jk\alpha\beta} = 1, \quad \forall jk \in \mathcal{E}; \quad (2.2)$$

$$\sum_{\alpha \in \mathcal{D}_j} z_{jk\alpha\beta} = z_{k\beta}, \quad \forall jk \in \mathcal{E}, \beta \in \mathcal{D}_k; \quad \sum_{\beta \in \mathcal{D}_k} z_{jk\alpha\beta} = z_{j\alpha}, \quad \forall jk \in \mathcal{E}, \alpha \in \mathcal{D}_j, \quad (2.3)$$

where (2.2) expresses normalization constraints and (2.3) captures marginalization constraints. This LP has integral optimal solutions if \mathcal{E} is a forest (Chekuri et al., 2001; Wainwright et al., 2002; Chekuri et al., 2005). In networks of general topology, however, the optimal solution can be fractional (as expected, since the problem is NP-hard). Other important exceptions can be found, however, specifically by focusing on constraints on the potentials rather than constraints on the topology. We discuss one such example in the following section.

2.2.2 Markov networks with submodular potentials

We consider a special class of Markov networks, common in vision applications, in which inference reduces to a tractable min-cut problem (Greig et al., 1989; Kolmogorov and Zabih, 2004). We assume that (1) all variables are binary ($\mathcal{D}_j = \{0, 1\}$), and (2) all edge potentials are “regular” (i.e., submodular):

$$\log \phi_{jk}(0, 0) + \log \phi_{jk}(1, 1) \geq \log \phi_{jk}(1, 0) + \log \phi_{jk}(0, 1), \quad \forall jk \in \mathcal{E}. \quad (2.4)$$

Such potentials prefer assignments where connected nodes have the same label, that is, $y_j = y_k$. This notion of regularity can be extended to potentials over more than two variables (Kolmogorov and Zabih, 2004). These assumptions ensure that the LP in Eq. (2.1) has integral optimal solutions (Chekuri et al., 2001; Kolmogorov and Wainwright, 2005; Chekuri et al., 2005). Similar kinds of networks (defined also for non-binary variables and non-pairwise potentials) were called “associative Markov networks” by Taskar et al. (2004a) and Anguelov et al. (2005), who used them for object segmentation and hypertext classification.

In figure-ground segmentation (see Fig. 2.1a), the node potentials capture local evidence about the label of a pixel or range scan point. Edges usually connect nearby pixels in an image, and serve to correlate their labels. Assuming that such correlations tend to be *positive* (connected nodes tend to have the same label) leads us to consider simplified edge potentials of the form $\phi_{jk}(y_j, y_k) = \exp\{-s_{jk}\mathbb{I}(y_j \neq y_k)\}$, where s_{jk} is a nonnegative penalty for assigning y_j and y_k different labels. Note that such potentials are regular if $s_{jk} \geq 0$. Expressing node potentials as $\phi_j(y_j) = \exp\{s_j y_j\}$, we have $P(\mathbf{y}) \propto \exp\left\{\sum_{j \in \mathcal{V}} s_j y_j - \sum_{jk \in \mathcal{E}} s_{jk} \mathbb{I}(y_j \neq y_k)\right\}$. Under this restriction on

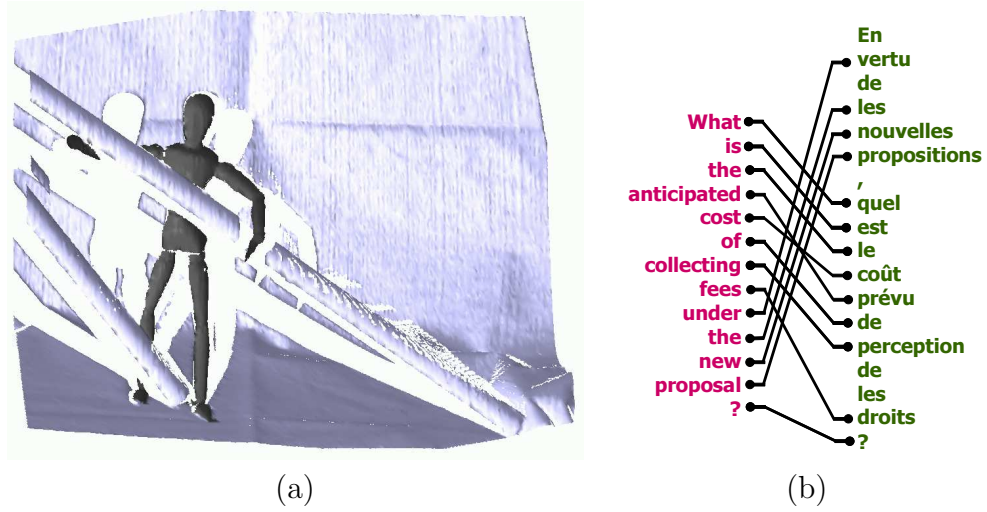


Figure 2.1: Structured prediction applications: (a) 3D figure-ground segmentation; (b) Word alignment in machine translation.

the potentials, we can obtain the following (simpler) LP:

$$\begin{aligned}
 \max_{0 \leq \mathbf{z} \leq 1} \quad & \sum_{j \in \mathcal{V}} s_j z_j - \sum_{jk \in \mathcal{E}} s_{jk} z_{jk} \\
 \text{s.t.} \quad & z_j - z_k \leq z_{jk}, \quad z_k - z_j \leq z_{jk}, \quad \forall jk \in \mathcal{E},
 \end{aligned} \tag{2.5}$$

where the continuous variables z_j correspond to a relaxation of the binary variables y_j , and the constraints encode $z_{jk} = \mathbb{I}(z_j \neq z_k)$. To see this, note that the constraints can be equivalently expressed as $|z_j - z_k| \leq z_{jk}$. Because s_{jk} is positive, $z_{jk} = |z_k - z_j|$ at the maximum, which is equivalent to $\mathbb{I}(z_j \neq z_k)$ if the z_j, z_k variables are binary. An integral optimal solution always exists, since the constraint matrix is totally unimodular (Schrijver, 2003), hence the relaxation is exact.

We can parameterize the node and edge potentials in terms of user-provided features \mathbf{x}_j and \mathbf{x}_{jk} associated with the nodes and edges. In particular, in 3D range data, \mathbf{x}_j might involve spin-image features or spatial occupancy histograms of a point j , while \mathbf{x}_{jk} might include the distance between points j and k , the dot-product of

their normals, etc. The simplest model of dependence is a linear combination of features: $s_j = \mathbf{w}_n^\top \mathbf{f}_n(\mathbf{x}_j)$ and $s_{jk} = \mathbf{w}_e^\top \mathbf{f}_e(\mathbf{x}_{jk})$, where \mathbf{w}_n and \mathbf{w}_e are node and edge parameters, and \mathbf{f}_n and \mathbf{f}_e are node and edge feature mappings, of dimension d_n and d_e , respectively. To ensure non-negativity of s_{jk} , we assume that the edge features \mathbf{f}_e are nonnegative and we impose the restriction $\mathbf{w}_e \geq 0$. This constraint is incorporated into the learning formulation we present below. We assume that the feature mappings \mathbf{f} are provided by the user and our goal is to estimate parameters \mathbf{w} from labeled data. We abbreviate the score assigned to a labeling \mathbf{y} for an input \mathbf{x} as $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_j y_j \mathbf{w}_n^\top \mathbf{f}_n(\mathbf{x}_j) - \sum_{jk \in \mathcal{E}} y_{jk} \mathbf{w}_e^\top \mathbf{f}_e(\mathbf{x}_{jk})$, where $y_{jk} = \mathbb{I}(y_j \neq y_k)$.

2.2.3 Matchings

Consider modeling the task of word alignment of parallel bilingual sentences (Fig. 2.1b) as a maximum weight bipartite matching problem in a graph, where the nodes $\mathcal{V} = \mathcal{V}^s \cup \mathcal{V}^t$ correspond to the words in the “source” sentence (\mathcal{V}^s) and the “target” sentence (\mathcal{V}^t) and the edges $\mathcal{E} = \{jk : j \in \mathcal{V}^s, k \in \mathcal{V}^t\}$ correspond to possible alignments between the words. For simplicity, we assume in this chapter that the source word aligns to at most one word (zero or one) in the other sentence — this constraint will be lifted in Chapter 3 to obtain a more realistic model. The edge weight s_{jk} represents the degree to which word j in one sentence can translate into the word k in the other sentence. Our objective is to find an alignment that maximizes the sum of edge scores. We represent a matching using a set of binary variables y_{jk} that are set to 1 if word j is assigned to word k in the other sentence, and 0 otherwise. The score of an assignment is the sum of edge scores: $s(\mathbf{y}) = \sum_{jk \in \mathcal{E}} s_{jk} y_{jk}$. The maximum weight bipartite matching problem, $\arg \max_{\mathbf{y} \in \mathcal{Y}} s(\mathbf{y})$, can be found by

solving the following LP:

$$\begin{aligned}
 \max_{0 \leq \mathbf{z} \leq 1} \quad & \sum_{jk \in \mathcal{E}} s_{jk} z_{jk} & (2.6) \\
 \text{s.t.} \quad & \sum_{j \in \mathcal{V}^s} z_{jk} \leq 1, \quad \forall k \in \mathcal{V}^t; & \sum_{k \in \mathcal{V}^t} z_{jk} \leq 1, \quad \forall j \in \mathcal{V}^s.
 \end{aligned}$$

where again the continuous variables z_{jk} correspond to the relaxation of the binary variables y_{jk} . As in the min-cut problem, this LP is guaranteed to have integral solutions for any scoring function $s(\mathbf{y})$ (Schrijver, 2003).

For word alignment, the scores s_{jk} can be defined in terms of the word pair jk and input features associated with \mathbf{x}_{jk} . We can include the identity of the two words, the relative position in the respective sentences, the part-of-speech tags, the string similarity (for detecting cognates), etc. We let $s_{jk} = \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk})$ for a user-provided feature mapping \mathbf{f} and abbreviate $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{jk} y_{jk} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk})$.

2.2.4 General structure

More generally, we consider prediction problems in which the input $\mathbf{x} \in \mathcal{X}$ is an arbitrary structured object and the output is a vector of values $\mathbf{y} = (y_1, \dots, y_{L_{\mathbf{x}}})$ encoding, for example, a matching or a cut in the graph. We assume that the length $L_{\mathbf{x}}$ and the structure encoded by \mathbf{y} depend deterministically on the input \mathbf{x} . In our word alignment example, the output space is defined by the length of the two sentences. Denote the output space for a given input \mathbf{x} as $\mathcal{Y}(\mathbf{x})$ and the entire output space as $\mathcal{Y} = \bigcup_{\mathbf{x} \in \mathcal{X}} \mathcal{Y}(\mathbf{x})$.

Consider the class of structured prediction models \mathcal{H} defined by the linear family:

$$h_{\mathbf{w}}(\mathbf{x}) \doteq \arg \max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}), \quad (2.7)$$

where $\mathbf{f}(\mathbf{x}, \mathbf{y})$ is a vector of functions $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}^n$. This formulation is very general. Indeed, it is too general for our purposes—for many $(\mathbf{f}, \mathcal{Y})$ pairs, finding the optimal \mathbf{y} is intractable. We specialize to the class of models in which the optimization problem in Eq. (2.7) can be solved in polynomial time via convex optimization; this is still a very large class of models. Beyond the examples discussed here, it includes weighted context-free grammars and dependency grammars (Manning and Schütze, 1999) and string edit distance models for sequence alignment (Durbin et al., 1998).

2.3 Large margin estimation

We assume a set of training instances $S = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$, where each instance consists of a structured object \mathbf{x}_i (such as a graph) and a target solution \mathbf{y}_i (such as a matching). Consider learning the parameters \mathbf{w} in the conditional likelihood setting. We can define $P_{\mathbf{w}}(\mathbf{y} | \mathbf{x}) = \frac{1}{Z_{\mathbf{w}}(\mathbf{x})} \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})\}$, where $Z_{\mathbf{w}}(\mathbf{x}) = \sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{x})} \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}')\}$, and maximize the conditional log-likelihood $\sum_i \log P_{\mathbf{w}}(\mathbf{y}_i | \mathbf{x}_i)$, perhaps with additional regularization of the parameters \mathbf{w} . As we have noted earlier, however, the problem of computing the partition function $Z_{\mathbf{w}}(\mathbf{x})$ is computationally intractable for many of the problems we are interested in. In particular, it is #P-complete for matchings and min-cuts (Valiant, 1979; Jerrum and Sinclair, 1993).

We thus retreat from conditional likelihood and consider the max-margin formulation developed in several recent papers (Collins, 2002; Taskar et al., 2004b; Tsochantaridis et al., 2005). In this formulation, we seek to find parameters \mathbf{w} such that:

$$\mathbf{y}_i = h_{\mathbf{w}}(\mathbf{x}_i) = \arg \max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \mathbf{y}'_i), \quad \forall i,$$

where $\mathcal{Y}_i = \mathcal{Y}(\mathbf{x}_i)$. The solution space \mathcal{Y}_i depends on the structured object \mathbf{x}_i ; for example, the space of possible matchings depends on the precise set of nodes and

edges in the graph.

As in univariate prediction, we measure the error of prediction using a loss function $\ell(\mathbf{y}_i, \mathbf{y}'_i)$. To obtain a convex formulation, we upper bound the loss $\ell(\mathbf{y}_i, h_{\mathbf{w}}(\mathbf{x}_i))$ using the hinge function²: $L_H(\mathbf{w}) \doteq \max_{\mathbf{y}'_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i)]$, where $\ell_i(\mathbf{y}'_i) = \ell(\mathbf{y}_i, \mathbf{y}'_i)$, and $\mathbf{f}_i(\mathbf{y}'_i) = \mathbf{f}(\mathbf{x}_i, \mathbf{y}'_i)$. Minimizing this upper bound will force the true structure \mathbf{y}_i to be optimal with respect to \mathbf{w} for each instance i :

$$\min_{\mathbf{w} \in \mathcal{W}} \sum_i \max_{\mathbf{y}'_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i)] - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i), \quad (2.8)$$

where \mathcal{W} is the set of allowed parameters \mathbf{w} . We assume that the parameter space \mathcal{W} is a convex set, typically a norm ball $\{\mathbf{w} : \|\mathbf{w}\|_p \leq \gamma\}$ with $p = 1, 2$ and a regularization parameter γ . In the case that $\mathcal{W} = \{\mathbf{w} : \|\mathbf{w}\|_2 \leq \gamma\}$, this formulation is equivalent to the standard large margin formulation using slack variables ξ and slack penalty C (cf. [Taskar et al., 2004b](#)), for some suitable values of C depending on γ . The correspondence can be seen as follows: let $\mathbf{w}^*(C)$ be a solution to the optimization problem with slack penalty C and define $\gamma(C) = \|\mathbf{w}^*(C)\|$. Then \mathbf{w}^* is also a solution to Eq. (2.8). Conversely, we can invert the mapping $\gamma(\cdot)$ to find those values of C (possibly non-unique) that give rise to the same solution as Eq. (2.8) for a specific γ . In the case of submodular potentials, there are additional linear constraints on the edge potentials. In the setting of Eq. (2.5), the constraint $w_e \geq 0$ is sufficient. For more general submodular potentials, we can parameterize the log of the edge potential using four sets of edge parameters, $\mathbf{w}_{e00}, \mathbf{w}_{e01}, \mathbf{w}_{e10}, \mathbf{w}_{e11}$, as follows: $\log \phi_{jk}(\alpha, \beta) = \mathbf{w}_{e\alpha\beta}^\top \mathbf{f}(\mathbf{x}_{jk})$. Assuming, as before, that the edge features are nonnegative, the regularity of the potentials can be enforced via a linear constraint: $\mathbf{w}_{e00} + \mathbf{w}_{e11} \geq \mathbf{w}_{e10} + \mathbf{w}_{e01}$, where the inequality should be interpreted componentwise.

²Under the assumption that the true label belongs to the output space in our model, i.e. $\mathbf{y}_i \in \mathcal{Y}_i$ (and only if), then it not hard to show that the hinge loss upper bounds the true loss of our predictor: $L_H(\mathbf{w}) \geq \ell(\mathbf{y}_i, h_{\mathbf{w}}(\mathbf{x}_i))$.

The key to solving Eq. (2.8) efficiently is the *loss-augmented inference problem*,

$$\max_{\mathbf{y}'_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i)]. \quad (2.9)$$

This optimization problem has precisely the same form as the prediction problem whose parameters we are trying to learn— $\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i)$ —but with an additional term corresponding to the loss function. Tractability of the loss-augmented inference thus depends not only on the tractability of $\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i)$, but also on the form of the loss term $\ell_i(\mathbf{y}'_i)$. A natural choice in this regard is the Hamming distance, which simply counts the number of variables in which a candidate solution \mathbf{y}'_i differs from the target output \mathbf{y}_i . In general, we need only assume that the loss function decomposes over the variables in \mathbf{y}_i .

In particular, for word alignment, we use weighted Hamming distance, which counts the number of variables in which a candidate matching \mathbf{y}'_i differs from the target alignment \mathbf{y}_i , with different cost for false positives (c^+) and false negatives (c^-):

$$\begin{aligned} \ell(\mathbf{y}_i, \mathbf{y}'_i) &= \sum_{jk \in \mathcal{E}_i} [c^- y_{i,jk}(1 - y'_{i,jk}) + c^+ y'_{i,jk}(1 - y_{i,jk})] \\ &= \sum_{jk \in \mathcal{E}_i} c^- y_{i,jk} + \sum_{jk \in \mathcal{E}_i} [c^+ - (c^- + c^+) y_{i,jk}] y'_{i,jk}, \end{aligned} \quad (2.10)$$

where $y_{i,jk}$ indicates the presence of edge jk in example i and \mathcal{E}_i is the set of edges in example i . The loss-augmented matching problem can then be written as an LP similar to Eq. (2.6) (without the constant term $\sum_{jk} c^- y_{i,jk}$):

$$\begin{aligned} \max_{0 \leq \mathbf{z}_i \leq 1} \quad & \sum_{jk \in \mathcal{E}_i} z_{i,jk} [\mathbf{w}^\top \mathbf{f}(\mathbf{x}_{i,jk}) + c^+ - (c^- + c^+) y_{i,jk}] \\ \text{s.t.} \quad & \sum_{j \in \mathcal{V}_i^s} z_{i,jk} \leq 1, \quad \forall k \in \mathcal{V}_i^t; \quad \sum_{k \in \mathcal{V}_i^t} z_{i,jk} \leq 1, \quad \forall j \in \mathcal{V}_i^s, \end{aligned}$$

where $\mathbf{f}(\mathbf{x}_{i,jk})$ is the vector of features of the edge jk in example i and \mathcal{V}_i^s and \mathcal{V}_i^t are the nodes in example i . As before, the continuous variables $z_{i,jk}$ correspond to the binary values $y'_{i,jk}$.

Generally, suppose we can express the prediction problem as an LP:

$$\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) = \max_{\mathbf{z}_i \in \mathcal{Z}_i} \mathbf{w}^\top \mathbf{F}_i \mathbf{z}_i,$$

where

$$\mathcal{Z}_i = \{\mathbf{z}_i : \mathbf{A}_i \mathbf{z}_i \leq \mathbf{b}_i, 0 \leq \mathbf{z}_i \leq 1\}, \quad (2.11)$$

for appropriately defined \mathbf{F}_i , \mathbf{A}_i and \mathbf{b}_i . Then we have a similar LP for the loss-augmented inference for each example i :

$$\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i) = d_i + \max_{\mathbf{z}_i \in \mathcal{Z}_i} (\mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i)^\top \mathbf{z}_i, \quad (2.12)$$

for appropriately defined d_i and \mathbf{c}_i . For the matching case, $d_i = \sum_{jk} c^- y_{i,jk}$ is the constant term, \mathbf{F}_i is a matrix that has a column of features $\mathbf{f}(\mathbf{x}_{i,jk})$ for each edge jk in example i , and \mathbf{c}_i is the vector of the loss terms $c^+ - (c^- + c^+) y_{i,jk}$. Let $\mathbf{z} = \{\mathbf{z}_1, \dots, \mathbf{z}_m\}$ and $\mathcal{Z} = \mathcal{Z}_1 \times \dots \times \mathcal{Z}_m$. With these definitions, we have the following saddle-point problem:

$$\min_{\mathbf{w} \in \mathcal{W}} \max_{\mathbf{z} \in \mathcal{Z}} \sum_i (\mathbf{w}^\top \mathbf{F}_i \mathbf{z}_i + \mathbf{c}_i^\top \mathbf{z}_i - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i)) \quad (2.13)$$

where we have omitted the constant term $\sum_i d_i$. The only difference between this formulation and our initial formulation in Eq. (2.8) is that we have created a concise continuous optimization problem by replacing the discrete \mathbf{y}'_i 's with continuous \mathbf{z}_i 's.

When the prediction problem is intractable (for example, in general Markov networks or tripartite matchings), we can use a convex relaxation (for example, a linear

or semidefinite program) to upper bound $\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i)$ and obtain an approximate maximum-margin formulation. This is the approach taken in Taskar et al. (2004b) for general Markov networks using the LP in Eq. (2.1).

To solve (2.13), we could proceed by making use of Lagrangian duality. This approach, explored in Taskar et al. (2004a, 2005a), yields a joint convex optimization problem. If the parameter space \mathcal{W} is described by linear and convex quadratic constraints, the result is a convex quadratic program which can be solved using a generic QP solver.

We briefly outline this approach below, but in this chapter, we take a different approach, solving the problem in its natural saddle-point form. As we discuss in the following section, this approach allows us to exploit the structure of \mathcal{W} and \mathcal{Z} *separately*, allowing for efficient solutions for a wider range of parameterizations and structures. It also opens up alternatives with respect to numerical algorithms.

Before moving on to solution of the saddle-point problem, we consider the joint convex form when the feasible set has the form of (2.11) and the loss-augmented inference problem is a LP, as in (2.12). Using commercial convex optimization solvers for this formulation will provide us with a comparison point for our saddle-point solver. We now proceed to present this alternative form.

To transform the saddle-point form of (2.13) into a standard convex optimization form, we take the dual of the individual loss-augmented LPs (2.12):

$$\max_{\mathbf{z}_i \in \mathcal{Z}_i} (\mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i)^\top \mathbf{z}_i = \min_{(\lambda_i, \mu_i) \in \Lambda_i(\mathbf{w})} \mathbf{b}_i^\top \lambda_i + \mathbf{1}^\top \mu_i \quad (2.14)$$

where $\Lambda_i(\mathbf{w}) = \{(\lambda_i, \mu_i) \geq 0 : \mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i \leq \mathbf{A}_i^\top \lambda_i + \mu_i\}$ defines the feasible space for the dual variables λ_i and μ_i . Substituting back in equation (2.13) and writing

$\lambda = (\lambda_1, \dots, \lambda_m)$, $\mu = (\mu_1, \dots, \mu_m)$, we obtain (omitting the constant $\sum_i d_i$):

$$\begin{aligned} \min_{\mathbf{w} \in \mathcal{W}, (\lambda, \mu) \geq 0} \quad & \sum_i (\mathbf{b}_i^\top \lambda_i + \mathbf{1}^\top \mu_i - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i)) \\ \text{s.t.} \quad & \mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i \leq \mathbf{A}_i^\top \lambda_i + \mu_i \quad i = 1, \dots, m. \end{aligned} \quad (2.15)$$

If \mathcal{W} is defined by linear and convex quadratic constraints, the above optimization problem can be solved using standard commercial solvers. The number of variables and constraints in this problem is linear in the number of the parameters *and* the training data (for example nodes and edges). On the other hand, note that the constraints in equation (2.15) intermix the structure of \mathcal{W} and \mathcal{Z} together. Hence, even though each original constraint set could have a simple form (e.g. a ball for \mathcal{W} and a product of matchings for \mathcal{Z}), the resulting optimization formulation (2.15) makes it hard to take advantage of this structure. By contrast, the saddle-point formulation (2.13) preserves the simple structure of the constraint sets. We will see in the next section how this can be exploited to obtain an efficient optimization algorithm.

2.4 Saddle-point problems and the dual extragradient method

We begin by establishing some notation and definitions. Denote the objective of the saddle-point problem in (2.13) by:

$$\mathcal{L}(\mathbf{w}, \mathbf{z}) \doteq \sum_i \mathbf{w}^\top \mathbf{F}_i \mathbf{z}_i + \mathbf{c}_i^\top \mathbf{z}_i - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i).$$

$\mathcal{L}(\mathbf{w}, \mathbf{z})$ is bilinear in \mathbf{w} and \mathbf{z} , with gradient given by: $\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}, \mathbf{z}) = \sum_i \mathbf{F}_i \mathbf{z}_i - \mathbf{f}_i(\mathbf{y}_i)$ and $\nabla_{\mathbf{z}_i}\mathcal{L}(\mathbf{w}, \mathbf{z}) = \mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i$.

We can view this problem as a zero-sum game between two players, \mathbf{w} and \mathbf{z} . Consider a simple iterative improvement method based on gradient projections:

$$\mathbf{w}^{t+1} = \pi_{\mathcal{W}}(\mathbf{w}^t - \eta \nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}^t, \mathbf{z}^t)); \quad \mathbf{z}_i^{t+1} = \pi_{\mathcal{Z}_i}(\mathbf{z}_i^t + \eta \nabla_{\mathbf{z}_i}\mathcal{L}(\mathbf{w}^t, \mathbf{z}^t)), \quad (2.16)$$

where η is a step size and

$$\pi_{\mathcal{V}}(\mathbf{v}) \doteq \arg \min_{\mathbf{v}' \in \mathcal{V}} \|\mathbf{v} - \mathbf{v}'\|_2$$

denotes the Euclidean projection of a vector \mathbf{v} onto a convex set \mathcal{V} . In this simple iteration, each player makes a small best-response improvement without taking into account the effect of the change on the opponent's strategy. This usually leads to oscillations, and indeed, this method is generally not guaranteed to converge for bilinear objectives for any step size (Korpelevich, 1976; He and Liao, 2002). One way forward is to attempt to average the points $(\mathbf{w}^t, \mathbf{z}^t)$ to reduce oscillation. We pursue a different approach that is based on the dual extragradient method of Nesterov (2003). In our previous work (Taskar et al., 2006a), we used a related method, the extragradient method due to Korpelevich (1976). The dual extragradient is, however, a more flexible and general method, in terms of the types of projections and feasible sets that can be used, allowing a broader range of structured problems and parameter regularization schemes. Before we present the algorithm, we introduce some notation which will be useful for its description.

Let us combine \mathbf{w} and \mathbf{z} into a single vector, $\mathbf{u} = (\mathbf{w}, \mathbf{z})$, and define the joint feasible space $\mathcal{U} = \mathcal{W} \times \mathcal{Z}$. Note that \mathcal{U} is convex since it is a direct product of convex sets.

We denote the (affine) gradient operator on this joint space as

$$\begin{pmatrix} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \mathbf{z}) \\ -\nabla_{\mathbf{z}_1} \mathcal{L}(\mathbf{w}, \mathbf{z}) \\ \vdots \\ -\nabla_{\mathbf{z}_m} \mathcal{L}(\mathbf{w}, \mathbf{z}) \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & \mathbf{F}_1 & \cdots & \mathbf{F}_m \\ -\mathbf{F}_1^\top & & & \\ \vdots & & 0 & \\ -\mathbf{F}_m^\top & & & \end{pmatrix}}_{\mathbf{F}} \underbrace{\begin{pmatrix} \mathbf{w} \\ \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_m \end{pmatrix}}_{\mathbf{u}} - \underbrace{\begin{pmatrix} \sum_i \mathbf{f}_i(\mathbf{y}_i) \\ \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_m \end{pmatrix}}_{\mathbf{a}} = \mathbf{F}\mathbf{u} - \mathbf{a}.$$

2.4.1 Dual extragradient

We first present the dual extragradient algorithm of [Nesterov \(2003\)](#) using the Euclidean geometry induced by the standard 2-norm, and consider a non-Euclidean setup in [Sec. 2.4.2](#).

As shown in [Fig. 2.2](#), the dual extragradient algorithm proceeds using very simple gradient and projection calculations.

Initialize: Choose $\hat{\mathbf{u}} \in \mathcal{U}$, set $\mathbf{s}^{-1} = 0$.
 Iteration t , $0 \leq t \leq \tau$:

$$\begin{aligned} \mathbf{v} &= \pi_{\mathcal{U}}(\hat{\mathbf{u}} + \eta \mathbf{s}^{t-1}); \\ \mathbf{u}^t &= \pi_{\mathcal{U}}(\mathbf{v} - \eta(\mathbf{F}\mathbf{v} - \mathbf{a})); \\ \mathbf{s}^t &= \mathbf{s}^{t-1} - (\mathbf{F}\mathbf{u}^t - \mathbf{a}). \end{aligned} \tag{2.17}$$

Output: $\bar{\mathbf{u}}^\tau = \frac{1}{\tau+1} \sum_{t=0}^{\tau} \mathbf{u}^t$.

Figure 2.2: Euclidean dual extragradient algorithm.

To relate this generic algorithm to our setting, recall that \mathbf{u} is composed of subvectors \mathbf{w} and \mathbf{z} ; this induces a commensurate decomposition of the \mathbf{v} and \mathbf{s} vectors into subvectors. To refer to these subvectors we will abuse notation and use the symbols \mathbf{w} and \mathbf{z}_i as indices. Thus, we write $\mathbf{v} = (\mathbf{v}_{\mathbf{w}}, \mathbf{v}_{\mathbf{z}_1}, \dots, \mathbf{v}_{\mathbf{z}_m})$, and similarly for \mathbf{u} and \mathbf{s} . Using this notation, the generic algorithm in [Eq. \(2.17\)](#) expands into the following

dual extragradient algorithm for structured prediction (where the brackets represent gradient vectors):

$$\begin{aligned}
 \mathbf{v}_{\mathbf{w}} &= \pi_{\mathcal{W}}(\hat{\mathbf{u}}_{\mathbf{w}} + \eta \mathbf{s}_{\mathbf{w}}^{t-1}); & \mathbf{v}_{\mathbf{z}_i} &= \pi_{\mathcal{Z}_i}(\hat{\mathbf{u}}_{\mathbf{z}_i} + \eta \mathbf{s}_{\mathbf{z}_i}^{t-1}), \quad \forall i; \\
 \mathbf{u}_{\mathbf{w}}^t &= \pi_{\mathcal{W}}(\mathbf{v}_{\mathbf{w}} - \eta \left[\sum_i \mathbf{F}_i \mathbf{v}_{\mathbf{z}_i} - \mathbf{f}_i(\mathbf{y}_i) \right]); & \mathbf{u}_{\mathbf{z}_i}^t &= \pi_{\mathcal{Z}_i}(\mathbf{v}_{\mathbf{z}_i} + \eta [\mathbf{F}_i^\top \mathbf{v}_{\mathbf{w}} + \mathbf{c}_i]), \quad \forall i; \\
 \mathbf{s}_{\mathbf{w}}^t &= \mathbf{s}_{\mathbf{w}}^{t-1} - \left[\sum_i \mathbf{F}_i \mathbf{u}_{\mathbf{z}_i}^t - \mathbf{f}_i(\mathbf{y}_i) \right]; & \mathbf{s}_{\mathbf{z}_i}^t &= \mathbf{s}_{\mathbf{z}_i}^{t-1} + [\mathbf{F}_i^\top \mathbf{u}_{\mathbf{w}}^t + \mathbf{c}_i], \quad \forall i.
 \end{aligned}$$

In the convergence analysis of the dual extragradient (Nesterov, 2003), the stepsize η is set to the inverse of the Lipschitz constant (with respect to the 2-norm) of the gradient operator:

$$1/\eta = L \doteq \max_{\mathbf{u}, \mathbf{u}' \in \mathcal{U}} \frac{\|\mathbf{F}(\mathbf{u} - \mathbf{u}')\|_2}{\|\mathbf{u} - \mathbf{u}'\|_2} \leq \|\mathbf{F}\|_2,$$

where $\|\mathbf{F}\|_2$ is the largest singular value of the matrix \mathbf{F} . In practice, various simple heuristics can be considered for setting the stepsize, including search procedures based on optimizing the gap merit function (see, e.g., He and Liao, 2002).

2.4.1.1 Convergence

One measure of quality of a saddle-point solution is via the gap function:

$$\mathcal{G}(\mathbf{w}, \mathbf{z}) = \left[\max_{\mathbf{z}' \in \mathcal{Z}} \mathcal{L}(\mathbf{w}, \mathbf{z}') - \mathcal{L}^* \right] + \left[\mathcal{L}^* - \min_{\mathbf{w}' \in \mathcal{W}} \mathcal{L}(\mathbf{w}', \mathbf{z}) \right], \quad (2.18)$$

where the optimal loss is denoted $\mathcal{L}^* = \min_{\mathbf{w}' \in \mathcal{W}} \max_{\mathbf{z} \in \mathcal{Z}} \mathcal{L}(\mathbf{w}', \mathbf{z})$. For non-optimal points (\mathbf{w}, \mathbf{z}) , the gap $\mathcal{G}(\mathbf{w}, \mathbf{z})$ is positive and serves as a useful merit function, a measure of accuracy of a solution found by the extragradient algorithm. At an optimum

we have

$$\mathcal{G}(\mathbf{w}^*, \mathbf{z}^*) = \max_{\mathbf{z}' \in \mathcal{Z}} \mathcal{L}(\mathbf{w}^*, \mathbf{z}') - \min_{\mathbf{w}' \in \mathcal{W}} \mathcal{L}(\mathbf{w}', \mathbf{z}^*) = 0.$$

Define the Euclidean divergence function as

$$d(\mathbf{v}, \mathbf{v}') = \frac{1}{2} \|\mathbf{v} - \mathbf{v}'\|_2^2,$$

and define a restricted gap function parameterized by positive divergence radii $D_{\mathbf{w}}$ and $D_{\mathbf{z}}$

$$\mathcal{G}_{D_{\mathbf{w}}, D_{\mathbf{z}}}(\mathbf{w}, \mathbf{z}) = \max_{\mathbf{z}' \in \mathcal{Z}} \{\mathcal{L}(\mathbf{w}, \mathbf{z}') : d(\hat{\mathbf{z}}, \mathbf{z}') \leq D_{\mathbf{z}}\} - \min_{\mathbf{w}' \in \mathcal{W}} \{\mathcal{L}(\mathbf{w}', \mathbf{z}) : d(\hat{\mathbf{w}}, \mathbf{w}') \leq D_{\mathbf{w}}\},$$

where the point $\hat{\mathbf{u}} = (\hat{\mathbf{u}}_{\mathbf{w}}, \hat{\mathbf{u}}_{\mathbf{z}}) \in \mathcal{U}$ is an arbitrary point that can be thought of as the “center” of \mathcal{U} . Assuming there exists a solution $\mathbf{w}^*, \mathbf{z}^*$ such that $d(\hat{\mathbf{w}}, \mathbf{w}^*) \leq D_{\mathbf{w}}$ and $d(\hat{\mathbf{z}}, \mathbf{z}^*) \leq D_{\mathbf{z}}$, this restricted gap function coincides with the unrestricted function defined in Eq. (2.18). The choice of the center point $\hat{\mathbf{u}}$ should reflect an expectation of where the “average” solution lies, as will be evident from the convergence guarantees presented below. For example, we can take $\hat{\mathbf{u}}_{\mathbf{w}} = 0$ and let $\hat{\mathbf{u}}_{\mathbf{z}_i}$ correspond to the encoding of the target \mathbf{y}_i .

By Theorem 2 of [Nesterov \(2003\)](#), after τ iterations, the gap of $(\bar{\mathbf{w}}^\tau, \bar{\mathbf{z}}^\tau) = \bar{\mathbf{u}}^\tau$ is upper bounded by:

$$\mathcal{G}_{D_{\mathbf{w}}, D_{\mathbf{z}}}(\bar{\mathbf{w}}^\tau, \bar{\mathbf{z}}^\tau) \leq \frac{(D_{\mathbf{w}} + D_{\mathbf{z}})L}{\tau + 1}. \quad (2.19)$$

This implies that $\mathcal{O}(\frac{1}{\epsilon})$ steps are required to achieve a desired accuracy of solution ϵ as measured by the gap function. Note that the exponentiated gradient algorithm ([Bartlett et al., 2005](#)) has the same $\mathcal{O}(\frac{1}{\epsilon})$ convergence rate. This sublinear convergence rate is slow compared to interior point methods, which enjoy superlinear convergence ([Boyd and Vandenberghe, 2004](#)). However, the simplicity of each itera-

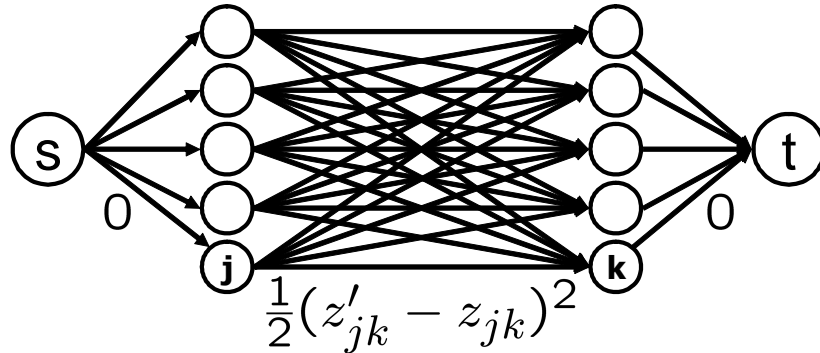


Figure 2.3: Euclidean projection onto the matching polytope using min-cost quadratic flow. Source s is connected to all the “source” nodes and target t connected to all the “target” nodes, using edges of capacity 1 and cost 0. The original edges jk have a quadratic cost $\frac{1}{2}(z'_{jk} - z_{jk})^2$ and capacity 1.

tion, the locality of key operations (projections), and the linear memory requirements make this a practical algorithm when the desired accuracy ϵ is not too small, and, in particular, these properties align well with the desiderata of large-scale machine learning algorithms. We illustrate these properties experimentally in Section 2.6.

2.4.1.2 Projections

The efficiency of the algorithm hinges on the computational complexity of the Euclidean projection onto the feasible sets \mathcal{W} and \mathcal{Z}_i . In the case of \mathcal{W} , projections are cheap when we have a 2-norm ball $\{\mathbf{w} : \|\mathbf{w}\|_2 \leq \gamma\}$: $\pi_{\mathcal{W}}(\mathbf{w}) = \gamma\mathbf{w} / \max(\gamma, \|\mathbf{w}\|_2)$. Additional non-negativity constraints on the parameters (e.g., $\mathbf{w}_e \geq 0$) can also be easily incorporated by clipping negative values. Projections onto the 1-norm ball are not expensive either (Boyd and Vandenberghe, 2004), but may be better handled by the non-Euclidean setup we discuss below.

We turn to the consideration of the projections onto \mathcal{Z}_i . The complexity of these projections is the key issue determining the viability of the extragradient approach

for our class of problems. In fact, for both alignment and matchings these projections turn out to reduce to classical network flow problems for which efficient solutions exist. In case of alignment, \mathcal{Z}_i is the convex hull of the bipartite matching polytope and the projections onto \mathcal{Z}_i reduce to the much-studied minimum cost quadratic flow problem (Bertsekas, 1998). In particular, the projection problem $\mathbf{z} = \pi_{\mathcal{Z}_i}(\mathbf{z}'_i)$ can be computed by solving

$$\begin{aligned} \min_{0 \leq \mathbf{z}_i \leq 1} \quad & \sum_{jk \in \mathcal{E}_i} \frac{1}{2} (z'_{i,jk} - z_{i,jk})^2 \\ \text{s.t.} \quad & \sum_{j \in \mathcal{V}_i^s} z_{i,jk} \leq 1, \quad \forall j \in \mathcal{V}_i^t; \quad \sum_{k \in \mathcal{V}_i^t} z_{i,jk} \leq 1, \quad \forall k \in \mathcal{V}_i^s. \end{aligned}$$

We use a standard reduction of bipartite matching to min-cost flow (see Fig. 2.3) by introducing a source node s connected to all the words in the “source” sentence, \mathcal{V}_i^s , and a target node t connected to all the words in the “target” sentence, \mathcal{V}_i^t , using edges of capacity 1 and cost 0. The original edges jk have a quadratic cost $\frac{1}{2}(z'_{i,jk} - z_{i,jk})^2$ and capacity 1. Since the edge capacities are 1, the flow conservation constraints at each original node ensure that the (possibly fractional) degree of each node in a valid flow is at most 1. Now the minimum cost flow from the source s to the target t computes projection of \mathbf{z}'_i onto \mathcal{Z}_i .

The reduction of the min-cut polytope projection to a convex network flow problem is more complicated; we present this reduction in Appendix 2.A. Algorithms for solving convex network flow problems (see, for example, Bertsekas et al., 1997) are nearly as efficient as those for solving linear min-cost flow problems, bipartite matchings and min-cuts. In case of word alignment, the running time scales with the cube of the sentence length. We use standard, publicly-available code for solving this problem (Guerriero and Tseng, 2002).³

³Available from http://www.math.washington.edu/~tseng/netflowg_nl/.

2.4.2 Non-Euclidean dual extragradient

Euclidean projections may not be easy to compute for many structured prediction problems or parameter spaces. The non-Euclidean version of the algorithm of [Nesterov \(2003\)](#) affords flexibility to use other types of (Bregman) projections. The basic idea is as follows. Let $d(\mathbf{u}, \mathbf{u}')$ denote a suitable divergence function (see below for a definition) and define a proximal step operator:

$$T_\eta(\mathbf{u}, \mathbf{s}) \doteq \arg \max_{\mathbf{u}' \in \mathcal{U}} [\mathbf{s}^\top (\mathbf{u}' - \mathbf{u}) - \frac{1}{\eta} d(\mathbf{u}, \mathbf{u}')].$$

Intuitively, the operator tries to make a large step from \mathbf{u} in the direction of \mathbf{s} but not too large as measured by $d(\cdot, \cdot)$. Then the only change to the algorithm is to switch from using a Euclidean projection of a gradient step $\pi_{\mathcal{U}}(\mathbf{u} + \frac{1}{\eta} \mathbf{s})$ to a proximal step in a direction of the gradient $T_\eta(\mathbf{u}, \mathbf{s})$ (see Fig. 2.4):

Initialize: Choose $\hat{\mathbf{u}} \in \tilde{\mathcal{U}}$, set $\mathbf{s}^{-1} = 0$.
 Iteration t , $0 \leq t \leq \tau$:

$\mathbf{v}_{\mathbf{w}} = T_\eta(\hat{\mathbf{u}}_{\mathbf{w}}, \mathbf{s}_{\mathbf{w}}^{t-1});$	$\mathbf{v}_{\mathbf{z}_i} = T_\eta(\hat{\mathbf{u}}_{\mathbf{z}_i}, \mathbf{s}_{\mathbf{z}_i}^{t-1}), \quad \forall i;$
$\mathbf{u}_{\mathbf{w}}^t = T_\eta(\mathbf{v}_{\mathbf{w}}, - \left[\sum_i \mathbf{F}_i \mathbf{v}_{\mathbf{z}_i} - \mathbf{f}_i(\mathbf{y}_i) \right]);$	$\mathbf{u}_{\mathbf{z}_i}^t = T_\eta(\mathbf{v}_{\mathbf{z}_i}, [\mathbf{F}_i^\top \mathbf{v}_{\mathbf{w}} + \mathbf{c}_i]), \quad \forall i;$
$\mathbf{s}_{\mathbf{w}}^t = \mathbf{s}_{\mathbf{w}}^{t-1} - \left[\sum_i \mathbf{F}_i \mathbf{u}_{\mathbf{z}_i}^t - \mathbf{f}_i(\mathbf{y}_i) \right];$	$\mathbf{s}_{\mathbf{z}_i}^t = \mathbf{s}_{\mathbf{z}_i}^{t-1} + [\mathbf{F}_i^\top \mathbf{u}_{\mathbf{w}}^t + \mathbf{c}_i], \quad \forall i.$

Output: $\bar{\mathbf{u}}^\tau = \frac{1}{\tau+1} \sum_{t=0}^{\tau} \mathbf{u}^t.$

Figure 2.4: Non-Euclidean dual extragradient algorithm.

To define the range of possible divergence functions and to state convergence properties of the algorithm, we will need some more definitions. We follow the development of [Nesterov \(2003\)](#). Given a norm $\|\cdot\|_{\mathcal{W}}$ on \mathcal{W} and norms $\|\cdot\|_{\mathcal{Z}_i}$ on \mathcal{Z}_i , we

combine them into a norm on \mathcal{U} as

$$\|\mathbf{u}\| = \max(\|\mathbf{w}\|^\mathcal{W}, \|\mathbf{z}_1\|^\mathcal{Z}_1, \dots, \|\mathbf{z}_m\|^\mathcal{Z}_m).$$

We denote the dual of \mathcal{U} (the vector space of linear functions on \mathcal{U}) as \mathcal{U}^* . The norm $\|\cdot\|$ on the space \mathcal{U} induces the dual norm $\|\cdot\|_*$ for all $\mathbf{s} \in \mathcal{U}^*$:

$$\|\mathbf{s}\|_* \doteq \max_{\mathbf{u} \in \mathcal{U}, \|\mathbf{u}\| \leq 1} \mathbf{s}^\top \mathbf{u}.$$

The Lipschitz constant with respect to this norm (used to set $\eta = 1/L$) is

$$L \doteq \max_{\mathbf{u}, \mathbf{u}' \in \mathcal{U}} \frac{\|\mathbf{F}(\mathbf{u} - \mathbf{u}')\|_*}{\|\mathbf{u} - \mathbf{u}'\|}.$$

The dual extragradient algorithm adjusts to the geometry induced by the norm by making use of Bregman divergences. Given a differentiable strongly convex function $h(\mathbf{u})$, we can define the Bregman divergence as:

$$d(\mathbf{u}, \mathbf{u}') = h(\mathbf{u}') - h(\mathbf{u}) - \nabla h(\mathbf{u})^\top (\mathbf{u}' - \mathbf{u}). \quad (2.20)$$

We need to be more specific about a few conditions to get a proper definition. First of all, the function $h(\mathbf{u})$ is *strongly convex* if and only if:

$$h(\alpha \mathbf{u} + (1 - \alpha) \mathbf{u}') \leq \alpha h(\mathbf{u}) + (1 - \alpha) h(\mathbf{u}') - \alpha(1 - \alpha) \frac{\sigma}{2} \|\mathbf{u} - \mathbf{u}'\|^2, \quad \forall \mathbf{u}, \mathbf{u}', \alpha \in [0, 1],$$

for some $\sigma > 0$, called the *convexity parameter* of $h(\cdot)$. In our setup, this function can be constructed from strongly convex functions on each of the spaces \mathcal{W} and \mathcal{Z}_i

by a simple sum: $h(\mathbf{u}) = h(\mathbf{w}) + \sum_i h(\mathbf{z}_i)$. It has a *conjugate function* defined as:

$$h^*(\mathbf{s}) \doteq \max_{\mathbf{u} \in \mathcal{U}} [\mathbf{s}^\top \mathbf{u} - h(\mathbf{u})].$$

Since $h(\cdot)$ is strongly convex, $h^*(\mathbf{u})$ is well-defined and differentiable at any $\mathbf{s} \in \mathcal{U}^*$.

We define its gradient set:

$$\tilde{\mathcal{U}} \doteq \{\nabla h^*(\mathbf{s}) : \mathbf{s} \in \mathcal{U}^*\}.$$

We further assume that $h(\cdot)$ is differentiable at any $\mathbf{u} \in \tilde{\mathcal{U}}$. Given that it is also strongly convex, for any two points $\mathbf{u} \in \tilde{\mathcal{U}}$ and $\mathbf{u}' \in \mathcal{U}$, we have

$$h(\mathbf{u}') \geq h(\mathbf{u}) + \nabla h(\mathbf{u})^\top (\mathbf{u}' - \mathbf{u}) + \frac{\sigma}{2} \|\mathbf{u}' - \mathbf{u}\|^2,$$

which shows that the Bregman divergence is always positive. Moreover, it motivates the definition of the Bregman divergence (2.20) between \mathbf{u}' and \mathbf{u} as the difference between $h(\mathbf{u}')$ and its linear approximation evaluated at \mathbf{u} . The stronger the curvature of $h(\cdot)$, the larger the divergence will scale with the Euclidean norm $\|\mathbf{u}' - \mathbf{u}\|^2$.

Note that when $\|\cdot\|$ is the 2-norm, we can use $h(\mathbf{u}) = \frac{1}{2} \|\mathbf{u}\|_2^2$, which has convexity parameter $\sigma = 1$, and induces the usual squared Euclidean distance $d(\mathbf{u}, \mathbf{u}') = \frac{1}{2} \|\mathbf{u} - \mathbf{u}'\|_2^2$. When $\|\cdot\|$ is the 1-norm, we can use the negative entropy $h(\mathbf{u}) = -H(\mathbf{u})$ (say if \mathcal{U} is a simplex), which also has $\sigma = 1$ and recovers the Kullback-Leibler divergence $d(\mathbf{u}, \mathbf{u}') = \text{KL}(\mathbf{u}' \|\mathbf{u})$.

With these definitions, the convergence bound in Eq. (2.19) applies to the non-Euclidean setup, but now the divergence radii are measured using Bregman divergence and the Lipschitz constant is computed with respect to a different norm.

Example 1: L_1 regularization

Suppose $\mathcal{W} = \{\mathbf{w} : \|\mathbf{w}\|_1 \leq \gamma\}$. We can transform this constraint set into a simplex constraint by the following variable transformation. Let $\mathbf{w} = \mathbf{w}^+ - \mathbf{w}^-$, $v_0 = 1 - \|\mathbf{w}\|_1/\gamma$, and $\mathbf{v} \doteq (v_0, \mathbf{w}^+/\gamma, \mathbf{w}^-/\gamma)$. Then $\mathcal{V} = \{\mathbf{v} : \mathbf{v} \geq 0; \mathbf{1}^\top \mathbf{v} = 1\}$ corresponds to \mathcal{W} . We define $h(\mathbf{v})$ as the negative entropy of \mathbf{v} :

$$h(\mathbf{v}) = \sum_d v_d \log v_d.$$

The resulting conjugate function and its gradient are given by

$$h^*(\mathbf{s}) = \log \sum_d e^{s_d}; \quad \frac{\partial h^*(\mathbf{s})}{\partial s_d} = \frac{e^{s_d}}{\sum_d e^{s_d}}.$$

Hence, the gradient space of $h^*(\mathbf{s})$ is the interior of the simplex, $\tilde{\mathcal{V}} = \{\mathbf{v} : \mathbf{v} > 0; \mathbf{1}^\top \mathbf{v} = 1\}$. The corresponding Bregman divergence is the standard Kullback-Leibler divergence

$$d(\mathbf{v}, \mathbf{v}') = \sum_d v'_d \log \frac{v'_d}{v_d}, \quad \forall \mathbf{v} \in \tilde{\mathcal{V}}, \mathbf{v}' \in \mathcal{V},$$

and the Bregman proximal step or projection, $\tilde{\mathbf{v}} = T_\eta(\mathbf{v}, \mathbf{s}) = \arg \max_{\mathbf{v}' \in \mathcal{V}} [\mathbf{s}^\top \mathbf{v}' - \frac{1}{\eta} d(\mathbf{v}, \mathbf{v}')]$ is given by a multiplicative update:

$$\tilde{v}_d = \frac{v_d e^{\eta s_d}}{\sum_d v_d e^{\eta s_d}}.$$

Note that we cannot choose $\hat{\mathbf{u}}_{\mathbf{v}} = (1, \mathbf{0}, \mathbf{0})$ as the center of $\tilde{\mathcal{V}}$ —given that the updates are multiplicative the algorithm will not make any progress in this case. In fact, this choice is precluded by the constraint that $\hat{\mathbf{u}}_{\mathbf{v}} \in \tilde{\mathcal{V}}$, not just $\hat{\mathbf{u}}_{\mathbf{v}} \in \mathcal{V}$. A reasonable choice is to set $\hat{\mathbf{u}}_{\mathbf{v}}$ to be the center of the simplex \mathcal{V} , $\hat{u}_{v_d} = \frac{1}{|\mathcal{V}|} = \frac{1}{2^{|\mathcal{W}|+1}}$.

Example 2: tree-structured marginals

Consider the case in which each example i corresponds to a tree-structured Markov network, and \mathcal{Z}_i is defined by the normalization and marginalization constraints in Eq. (2.2) and Eq. (2.3) respectively. These constraints define the space of valid marginals. For simplicity of notation, we assume that we are dealing with a single example i and drop the explicit index i . Let us use a more suggestive notation for the components of \mathbf{z} : $z_j(\alpha) = z_{j\alpha}$ and $z_{jk}(\alpha, \beta) = z_{jk\alpha\beta}$. We can construct a natural joint probability distribution via

$$P_{\mathbf{z}}(\mathbf{y}) = \prod_{jk \in \mathcal{E}} z_{jk}(y_j, y_k) \prod_{j \in \mathcal{V}} (z_j(y_j))^{1-q_j},$$

where q_j is the number of neighbors of node j . Now \mathbf{z} defines a point on the simplex of joint distributions over \mathcal{Y} , which has dimension $|\mathcal{Y}|$. One natural measure of complexity in this enlarged space is the 1-norm. We define $h(\mathbf{z})$ as the negative entropy of the distribution represented by \mathbf{z} :

$$h(\mathbf{z}) = \sum_{jk \in \mathcal{E}} \sum_{\alpha \in \mathcal{D}_j, \beta \in \mathcal{D}_k} z_{jk}(\alpha, \beta) \log z_{jk}(\alpha, \beta) + (1 - q_j) \sum_{j \in \mathcal{V}} \sum_{\alpha \in \mathcal{D}_j} z_j(\alpha) \log z_j(\alpha).$$

The resulting $d(\mathbf{z}, \mathbf{z}')$ is the Kullback-Leibler divergence $\text{KL}(P_{\mathbf{z}'} || P_{\mathbf{z}})$. The corresponding Bregman step or projection operator, $\tilde{\mathbf{z}} = T_{\eta}(\mathbf{z}, \mathbf{s}) = \arg \max_{\mathbf{z}' \in \mathcal{Z}} [\mathbf{s}^{\top} \mathbf{z}' - \frac{1}{\eta} \text{KL}(P_{\mathbf{z}'} || P_{\mathbf{z}})]$ is given by a multiplicative update on the space of distributions:

$$\begin{aligned} P_{\tilde{\mathbf{z}}}(\mathbf{y}) &= \frac{1}{Z} P_{\mathbf{z}}(\mathbf{y}) e^{\eta [\sum_{jk} s_{jk}(y_j, y_k) + \sum_j s_j(y_j)]} \\ &= \frac{1}{Z} \prod_{jk} z_{jk}(y_j, y_k) e^{\eta s_{jk}(y_j, y_k)} \prod_j (z_j(y_j))^{1-q_j} e^{\eta s_j(y_j)}, \end{aligned}$$

where we use the same indexing for the dual space vector \mathbf{s} as for \mathbf{z} and Z is a

normalization constant. Hence, to obtain the projection $\tilde{\mathbf{z}}$, we compute the node and edge marginals of the distribution $P_{\tilde{\mathbf{z}}}(\mathbf{y})$ via the standard sum-product dynamic programming algorithm using the node and edge potentials defined above. Note that the form of the multiplicative update of the projection resembles that of exponentiated gradient. As in the example above, we cannot let $\hat{\mathbf{u}}_{\mathbf{z}}$ be a corner (or any boundary point) of the simplex since $\tilde{\mathcal{Z}}$ does not include it. A reasonable choice for $\hat{\mathbf{u}}_{\mathbf{z}}$ would be either the center of the simplex or a point near the target structure but in the interior of the simplex.

2.5 Memory-efficient formulation

We now look at the memory requirements of the algorithm. The algorithm maintains the vector \mathbf{s}^τ as well as the running average, $\bar{\mathbf{u}}^\tau$, each with total dimensionality of $|\mathcal{W}| + |\mathcal{Z}|$. Note, however, that these vectors are related very simply by:

$$\mathbf{s}^\tau = - \sum_{t=0}^{\tau} (\mathbf{F}\mathbf{u}^t - \mathbf{a}) = -(\tau + 1)(\mathbf{F}\bar{\mathbf{u}}^\tau - \mathbf{a}).$$

So it suffices to only maintain the running average $\bar{\mathbf{u}}^\tau$ and reconstruct \mathbf{s} as needed.

In problems in which the number of examples m is large, we can take advantage of the fact that the memory needed to store the target structure \mathbf{y}_i is often much smaller than the corresponding vector \mathbf{z}_i . For example, for word alignment, we need $\mathcal{O}(|\mathcal{V}_i^s| \log |\mathcal{V}_i^t|)$ bits to encode a matching \mathbf{y}_i by using roughly $\log |\mathcal{V}_i^t|$ bits per node in \mathcal{V}_i^s to identify its match. By contrast, we need $|\mathcal{V}_i^s| |\mathcal{V}_i^t|$ floating numbers to maintain \mathbf{z}_i . The situation is worse in context-free parsing, where a parse tree \mathbf{y}_i requires space linear in the sentence length and logarithmic in grammar size, while $|\mathcal{Z}_i|$ is the product of the grammar size and the cube of the sentence length.

Note that from $\bar{\mathbf{u}}^\tau = (\bar{\mathbf{u}}_{\mathbf{w}}^\tau, \bar{\mathbf{u}}_{\mathbf{z}}^\tau)$, we only care about $\bar{\mathbf{u}}_{\mathbf{w}}^\tau$, the parameters of the

model, while the other component, $\bar{\mathbf{u}}_{\mathbf{z}}^t$, maintains the state of the algorithm. Fortunately, we can eliminate the need to store $\bar{\mathbf{u}}_{\mathbf{z}}$ by maintaining it implicitly, at the cost of storing a vector of size $|\mathcal{W}|$. This allows us to essentially have the same small memory footprint of online-type learning methods, where a single example is processed at a time and only a vector of parameters is maintained. In particular, instead of maintaining the entire vector $\bar{\mathbf{u}}^t$ and reconstructing \mathbf{s}^t from $\bar{\mathbf{u}}^t$, we can instead store only $\bar{\mathbf{u}}_{\mathbf{w}}^t$ and $\mathbf{s}_{\mathbf{w}}^t$ between iterations, since

$$\mathbf{s}_{\mathbf{z}_i}^t = (t + 1)(\mathbf{F}_i^\top \bar{\mathbf{u}}_{\mathbf{w}}^t + \mathbf{c}_i).$$

The diagram in Fig. 2.5 illustrates the process and the algorithm is summarized in Fig. 2.6. We use two “temporary” variables $\mathbf{v}_{\mathbf{w}}$ and $\mathbf{r}_{\mathbf{w}}$ of size $|\mathcal{W}|$ to maintain intermediate quantities. The additional vector $\mathbf{q}_{\mathbf{w}}$ shown in Fig. 2.5 is introduced only to allow the diagram to be drawn in a simplified manner; it can be eliminated by using $\mathbf{s}_{\mathbf{w}}$ to accumulate the gradients as shown in Fig. 2.6. The total amount of memory needed is thus four times the number of parameters plus memory for a single example $(\mathbf{v}_{\mathbf{z}_i}, \mathbf{u}_{\mathbf{z}_i})$. We assume that we do not need to store $\hat{\mathbf{u}}_{\mathbf{z}_i}$ explicitly but can construct it efficiently from $(\mathbf{x}_i, \mathbf{y}_i)$.

2.5.1 Kernels

Note that in case the dimensionality of the parameter space is much larger than the dimensionality of \mathcal{Z} , we can use a similar trick to only store variables of the size of \mathbf{z} . In fact, if $\mathcal{W} = \{\mathbf{w} : \|\mathbf{w}\|_2 \leq \gamma\}$ and we use Euclidean projections onto \mathcal{W} , we can exploit kernels to define infinite-dimensional feature spaces and derive a kernelized version of the algorithm.

For example, in the case of matchings of Sec. 2.2.3, we could define a kernel which

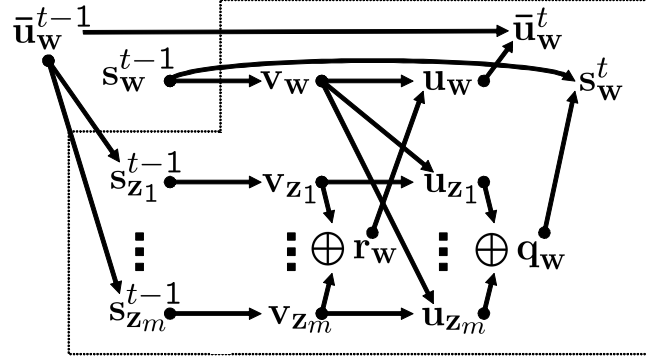


Figure 2.5: Dependency diagram for memory-efficient dual extragradient algorithm. The dotted box represents the computations of an iteration of the algorithm. Only $\bar{\mathbf{u}}_{\mathbf{w}}^t$ and $\mathbf{s}_{\mathbf{w}}^t$ are kept between iterations. Each example is processed one by one and the intermediate results are accumulated as $\mathbf{r}_{\mathbf{w}} = \mathbf{r}_{\mathbf{w}} - \mathbf{F}_i \mathbf{v}_{\mathbf{z}_i} + \mathbf{f}_i(\mathbf{y}_i)$ and $\mathbf{q}_{\mathbf{w}} = \mathbf{q}_{\mathbf{w}} - \mathbf{F}_i \mathbf{u}_{\mathbf{z}_i} + \mathbf{f}_i(\mathbf{y}_i)$. Details shown in Fig. 2.6, except that intermediate variables $\mathbf{u}_{\mathbf{w}}$ and $\mathbf{q}_{\mathbf{w}}$ are only used here for pictorial clarity.

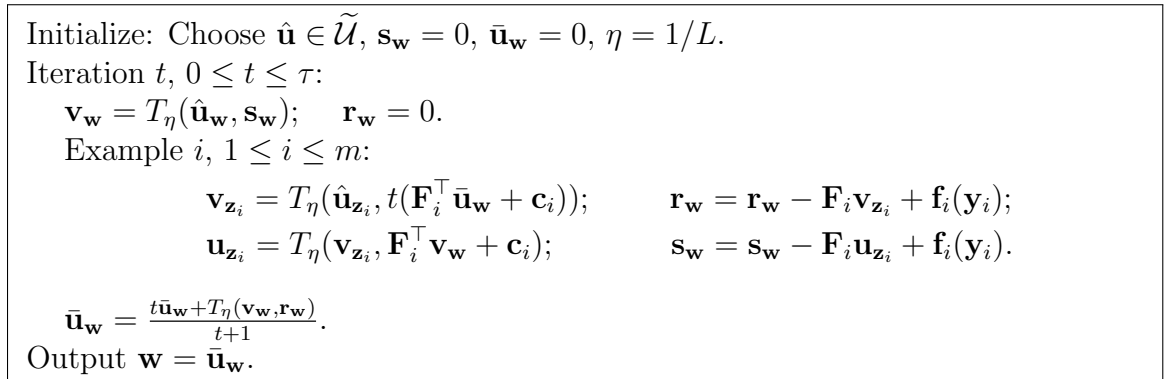


Figure 2.6: Memory-efficient dual extragradient algorithm.

depends on the features of an edge:

$$\mathbf{w} = \sum_l \alpha_l k(\mathbf{x}_l, \cdot),$$

where l ranges over all possible i, jk in the training set. Examples of quantities needed in the dual extragradient algorithm are the edge scores vector:

$$\mathbf{F}_i^\top \mathbf{v}_w = \left(\sum_l \alpha_l k(\mathbf{x}_l, \mathbf{x}_{i,jk}) \right)_{jk \in \mathcal{E}_i},$$

where α_l 's are derived from \mathbf{v}_w , and the dual step direction:

$$\begin{aligned} \mathbf{s}_w^t &= -(t+1) [\mathbf{F}_i \bar{\mathbf{u}}_{z_i}^t - \mathbf{f}_i(\mathbf{y}_i)] \\ &= -(t+1) \left[\sum_{i,jk} (\bar{u}_{i,jk}^t - y_{i,jk}) k(\mathbf{x}_{i,jk}, \cdot) \right], \end{aligned}$$

hence $\alpha_{i,jk} = (t+1)(y_{i,jk} - \bar{u}_{i,jk}^t)$ for \mathbf{s}_w^t . This direction step is then used for the Euclidean projection on the ball $\mathcal{W} = \{\mathbf{w} : \|\mathbf{w}\|_2 \leq \gamma\}$, which is easily done with kernels.

2.6 Experiments

In this section we describe experiments focusing on two of the structured models we described earlier: bipartite matchings for word alignments and restricted potential Markov nets for 3D segmentation.⁴ We compared three algorithms: the dual extragradient (`dual-ex`), the averaged projected gradient (`proj-grad`) defined in Eq. (2.16), and the averaged perceptron (Collins, 2002). For `dual-ex` and `proj-grad`, we used

⁴Software implementing our dual extragradient algorithm can be found at <http://www.cs.berkeley.edu/~slacoste/research/dualex>.

Euclidean projections, which can be formulated as min-cost quadratic flow problems. We used $\mathbf{w} = 0$ and \mathbf{z}_i corresponding to \mathbf{y}_i as the centroid $\hat{\mathbf{u}}$ in `dual-ex` and as the starting point of `proj-grad`.

In our experiments, we consider standard L_2 regularization, $\{\|\mathbf{w}\|_2 \leq \gamma\}$. A question which arises in practice is how to choose the regularization parameter γ . The typical approach is to run the algorithm for several values of the regularization parameter and pick the best model using a validation set. This can be quite expensive, though, and several recent papers have explored techniques for obtaining the whole regularization path, either exactly (Hastie et al., 2004), or approximately using path following techniques (Rosset). Instead, we run the algorithm without regularization ($\gamma = \infty$) and track its performance on the validation set, selecting the model with best performance. For comparison, whenever feasible with the available memory, we used commercial software to compute points on the regularization path. As we discuss below, the dual extragradient algorithm approximately follows the regularization path in our experiments (in terms of the training objective and test error) in the beginning and the end of the range of γ and often performs better in terms of generalization error in the mid-range.

2.6.1 Object segmentation

We tested our algorithm on a 3D scan segmentation problem using the class of Markov networks with regular potentials that were described in Section 2.2.2. The dataset is a challenging collection of cluttered scenes containing articulated wooden puppets (Anguelov et al., 2005). It contains eleven different single-view scans of three puppets of varying sizes and positions, with clutter and occluding objects such as rope, sticks and rings. Each scan consists of around 7,000 points. Our goal was to segment the scenes into two classes—`puppet` and `background`. We use five of the scenes

for our training data, three for validation and three for testing. Sample scans from the training and test set can be seen at <http://www.cis.upenn.edu/~taskar/3DSegment/>. We computed spin images of size 10×5 bins at two different resolutions, then scaled the values and performed PCA to obtain 45 principal components, which comprised our node features. We used the surface links output by the scanner as edges between points and for each edge only used a single feature, set to a constant value of 1 for all edges. This results in all edges having the same potential. The training data contains approximately 37,000 nodes and 88,000 edges. We used standard Hamming distance for our loss function $\ell(\mathbf{y}_i, \mathbf{y}'_i)$.

We compared the performance of the dual extragradient algorithm along its unregularized path to solutions of the regularized problems for different settings of the norm.⁵ For dual extragradient, the stepsize is set to $\eta = 1/\|\mathbf{F}\|_2 \approx 0.005$. We also compared to a variant of the averaged perceptron algorithm (Collins, 2002), where we use the batch updates to stabilize the algorithm, since we only have five training examples. We set the learning rate to 0.0007 by trying several values and picking the best value based on the validation data.

In Fig. 2.7(a) we track the hinge loss on the training data:

$$\sum_i \max_{\mathbf{y}'_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i)] - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i). \quad (2.21)$$

The hinge loss of the regularization path (**reg-path**) is the minimum loss for a given norm, and hence is always lower than the hinge loss of the other algorithms. However, as the norm increases and the model approaches the unregularized solution, **dual-ex** loss tends towards that of **reg-path**. Note that **proj-grad** behaves quite erratically in the range of the norms shown. Fig. 2.7(b) shows the growth of the norm as a function

⁵We used CPLEX to solve the regularized problems and also to find the projections onto the min-cut polytope, since the min-cost quadratic flow code we used (Guerrero and Tseng, 2002) does not support negative flows on edges, which are needed in the formulation presented in Appendix 2.A.

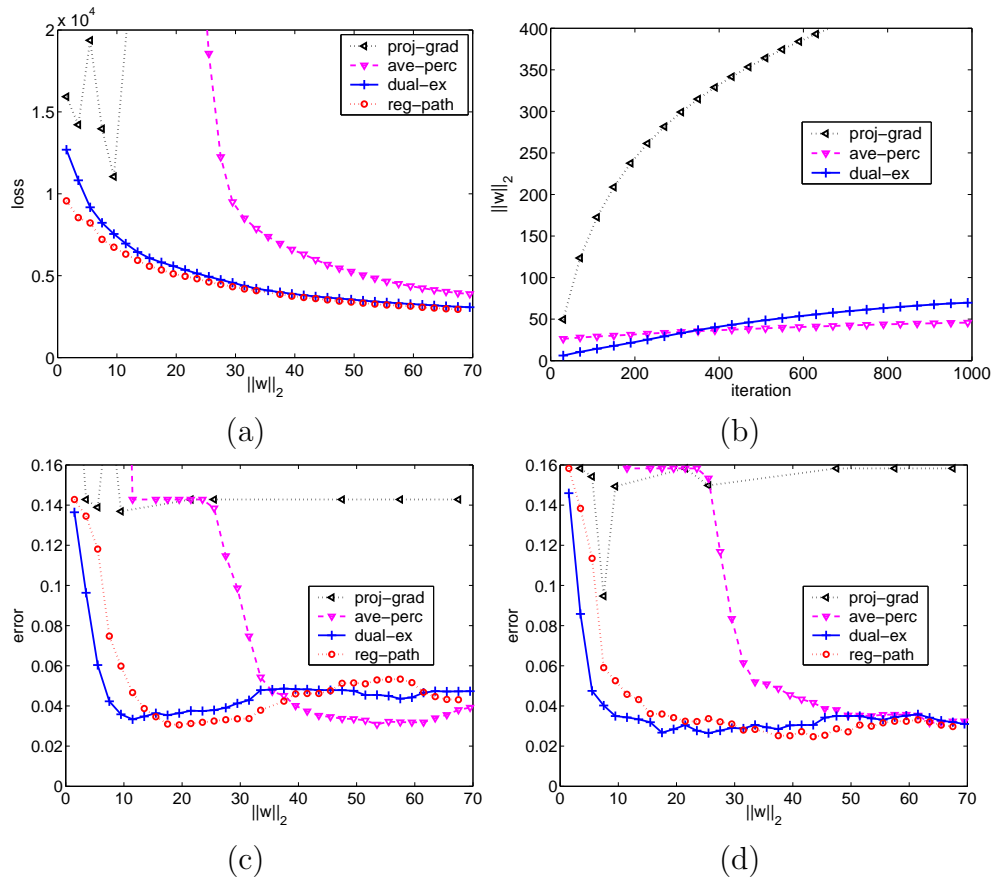


Figure 2.7: Object segmentation results: (a) Training hinge loss for the regularization path (reg-path), the averaged projected gradient (proj-grad), the averaged perceptron (ave-perc) and unregularized dual extragradient (dual-ex) vs. the norm of the parameters. (b) Norm of the parameters vs. iteration number for the three algorithms. (c) Validation error vs. the norm of the parameters. (d) Test error vs. the norm of the parameters.

of iteration number for `dual-ex` and `ave-perc`. The unregularized dual extragradient seems to explore the range of models (in terms on their norm) on the regularization path more thoroughly than the averaged perceptron and eventually asymptotes to the unregularized solution, while `proj-grad` quickly achieves very large norm.

Fig. 2.7(c) and Fig. 2.7(d) show validation and test error for the three algorithms. The best validation and test error achieved by the `dual-ex` and `ave-perc` algorithms as well as `reg-path` are fairly close, however, this error level is reached at very different norms. Since the number of scenes in the validation and test data is very small (three), because of variance, the best norm on validation is not very close to the best norm on the test set. Selecting the best model on the validation set leads to test errors of 3.4% for `dual-ex`, 3.5% for `ave-perc`, 3.6% for `reg-path` and 3.8% for `proj-grad` (`proj-grad` actually improves performance after the model norm is larger than 500, which is not shown in the graphs).

2.6.2 Word alignment

We also tested our algorithm on word-level alignment using a data set from the 2003 NAACL set (Mihalcea and Pedersen, 2003), the English-French Hansards task. This corpus consists of 1.1M pairs of sentences, and comes with a validation set of 37 sentence pairs and a test set of 447 word-aligned sentences. The validation and test sentences have been hand-aligned (see Och and Ney, 2003) and are marked with both *sure* and *possible* alignments. Using these alignments, the *alignment error rate* (AER) is calculated as:

$$AER(A, S, P) = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|},$$

where A is a set of proposed alignment pairs, S is the set of sure gold pairs, and P is the set of possible gold pairs (where $S \subseteq P$).

We experimented with two different training settings. In the first one, we split

the original test set into 100 training examples and 347 test examples—this dataset is called the ‘Gold’ dataset. In the second setting, we used GIZA++ (Och and Ney, 2003) to produce IBM Model 4 alignments for the unlabeled sentence pairs. We took the intersection of the predictions of the English-to-French and French-to-English Model 4 alignments on the first 5000 sentence pairs from the 1.1M sentences in order to experiment with the scaling of our algorithm (training on 500, 1000 and 5000 sentences). The number of edges for 500, 1000 and 5000 sentences of GIZA++ were about 31,000, 99,000 and 555,000 respectively. We still tested on the 347 Gold test examples, and used the validation set to select the stopping point. The stepsize for the dual extragradient algorithm was chosen to be $1/||\mathbf{F}||_2$.

We used statistics computed on the 1.1M sentence pairs as the edge features for our model. A detailed analysis of the constructed features and corresponding error analysis is presented in Taskar et al. (2005b). Example features include: a measure of mutual information between the two words computed from their co-occurrence in the aligned sentences (Dice coefficient); the difference between word positions; character-based similarity features designed to capture cognate (and exact match) information; and identity of the top five most frequently occurring words. We used the structured loss $\ell(\mathbf{y}_i, \mathbf{y}'_i)$ defined in Eq. (2.10) with $(c^+, c^-) = (1, 3)$ (where 3 was selected by testing several values on the validation set). We obtained low recall when using equal cost for both type of errors because the number of positive edges is significantly smaller than the number of negative edges, and so it is safe (precision-wise) for the model to predict fewer edges, hurting the recall. Increasing the cost for false negatives solves this problem.

Fig. 2.8(a) and Fig. 2.8(e) compare the hinge loss of the regularization path with the evolution of the objective for the unregularized dual extragradient, averaged projected gradient and averaged perceptron algorithms when trained on the Gold data

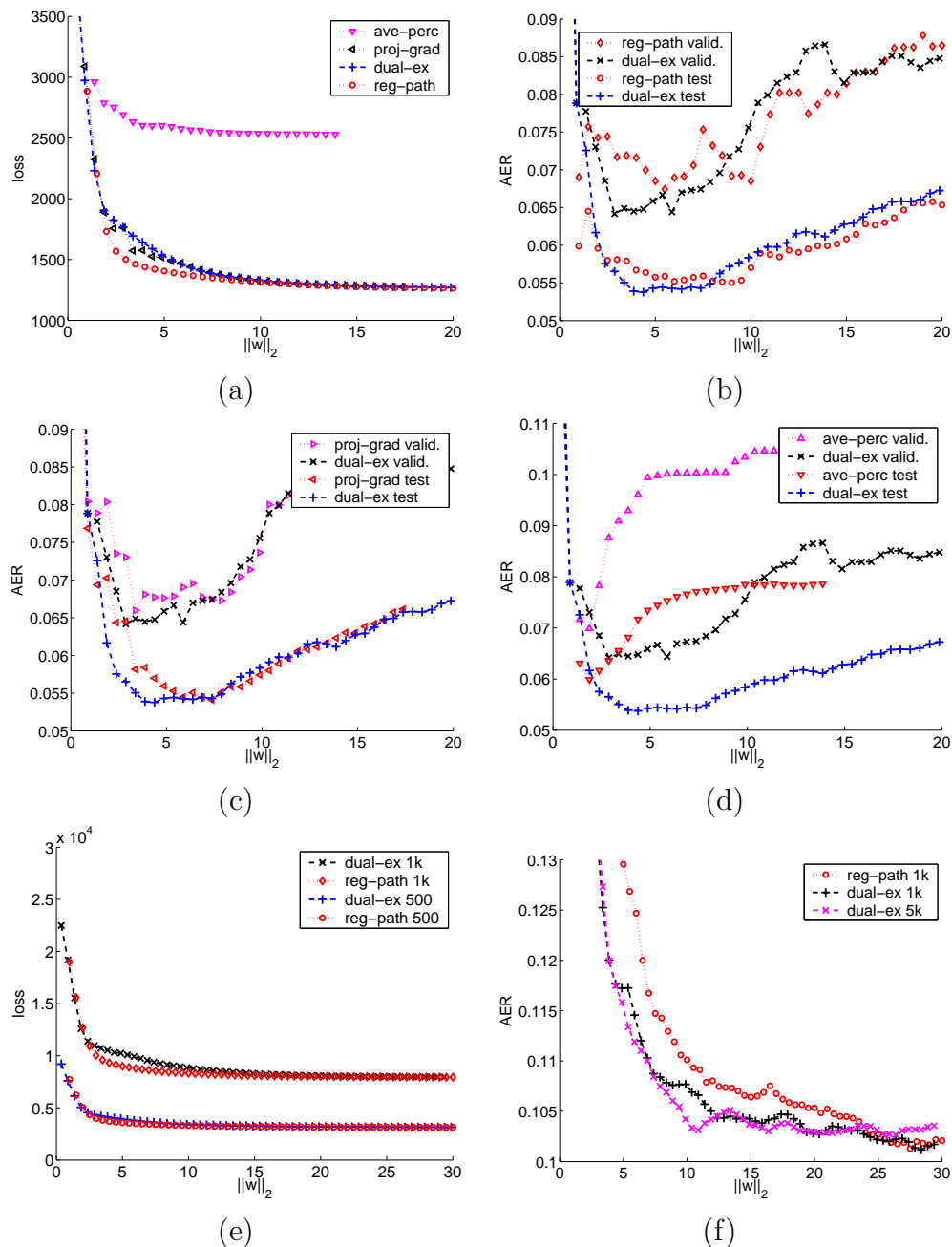


Figure 2.8: Word alignment results: (a) Training hinge loss for the three different algorithms and the regularization path on the Gold dataset. (b) AER for the unregularized dual extragradient (dual-ex) and the regularization path (reg-path) on the 347 Gold sentences (test) and the validation set (valid) when trained on the 100 Gold sentences; (c) Same setting as in (b), comparing dual-ex with the averaged projected-gradient (proj-grad); (d) Same setting as in (b), comparing proj-grad with the averaged perceptron (ave-perc); (e) Training hinge loss for dual-ex and reg-path on 500 and 1000 GIZA++ labeled sentences. (f) AER for dual-ex and reg-path tested on the Gold test set and trained on 1000 and 5000 GIZA++ sentences. The graph for 500 sentences is omitted for clarity.

set, 500 sentences and 1000 sentences of the GIZA++ output respectively.⁶ The dual extragradient path appears to follow the regularization path closely for $\|\mathbf{w}\| \leq 2$ and $\|\mathbf{w}\| \geq 12$. Fig. 2.8(b) compares the AER on the test set along the dual extragradient path trained on the Gold dataset versus the regularization path AER. The results on the validation set for each path are also shown. On the Gold data set, the minimum AER was reached roughly after 200 iterations.

Interestingly, the unregularized dual extragradient path seems to give better performance on the test set than that obtained by optimizing along the regularization path. The dominance of the dual extragradient path over the regularization path is more salient in Fig. 2.8(f) for the case where both are trained on 1000 sentences from the GIZA++ output. We conjecture that the dual extragradient method provides additional statistical regularization (compensating for the noisier labels of the GIZA++ output) by enforcing local smoothness of the path in parameter space.

The averaged projected gradient performed much better for this task than segmentation, getting somewhat close to the dual extragradient path as is shown in Fig. 2.8(c). The online version of the averaged perceptron algorithm varied significantly with the order of presentation of examples (up to five points of difference in AER between two orders). To alleviate this, we randomize the order of the points at each pass over the data. Fig. 2.8(d) shows that a typical run of averaged perceptron does somewhat worse than dual extragradient. The variance of the averaged perceptron performance for different datasets and learning rate choices was also significantly higher than for dual extragradient, which is more stable numerically. The online version of the averaged perceptron converged very quickly to its minimum AER score; converging in as few as five iterations for the Gold training set. Selecting the best model on the validation set leads to test errors of 5.6% for `dual-ex`, 5.6% for `reg-path`, 5.8% for

⁶ The regularization path is obtained by using the commercial optimization software Mosek with the QCQP formulation of Eq. (2.15). We did not obtain the path in the case of 5000 sentences, as Mosek runs out of memory.

proj-grad and 6.1% for ave-perc on the Gold data training set.

The running time for 500 iterations of dual extragradient on a 3.4 Ghz Intel Xeon CPU with 4G of memory was roughly 10 minutes, 30 minutes and 3 hours for 500, 1000 and 5000 sentences, respectively, showing the favorable linear scaling of the algorithm (linear in the number of edges). Note, by way of comparison, that Mosek ran out of memory for more than 1500 training sentences.

The framework we have presented here supports much richer models for word alignment; for example, allowing finer-grained, feature-based fertility control (number of aligned words for each word) as well as inclusion of positive correlations between adjacent edges in alignments. We present these extensions in Chapter 3.

2.7 Related work and discussion

We consider in this section related work which sheds additional light on some properties of our algorithm. The approach presented in this chapter relied on two main assumptions:

Tractable inference: That we can express the loss-augmented inference problem (2.12) as an LP of tractable size. This in turns imposes related assumptions of decomposability on the features $\mathbf{f}(\mathbf{x}, \mathbf{y})$ and the structured loss $\ell(\mathbf{y}, \mathbf{y}')$.

Tractable projections: That we can compute the (possibly non-Euclidean) projection on $\mathcal{W} \times \mathcal{Z}$ efficiently. We gave some examples in Sec. 2.4.2, but note that supposing that the first assumption is satisfied, this requires solving a QP in the worst case if we consider a Euclidean projection on a general polytope \mathcal{Z} (rather than the special cases we have considered in this paper for which the projection was much more efficient). In this case though, the gain over the polynomial QP formulation of equation (2.15) is less clear.

By comparison, the SVMstruct algorithm of [Tsochantaridis et al. \(2005\)](#) uses a cutting plane approach which needs only to solve the loss-augmented inference problem efficiently (to find the maximally violated constraint in their exponential-size QP which has the same solution as the large-margin objective of (2.15)). This inference problem could be an LP, a dynamic program, or something else. Their algorithm can thus in principle leverage more general losses and features. Moreover, [Finley and Joachims \(2008\)](#) provided theory on the behavior of the algorithm when the loss-augmented inference problem is solved approximately rather than exactly. On the other hand, their algorithm needs $\mathcal{O}(\frac{1}{\epsilon^2})$ iterations, in contrast to $\mathcal{O}(\frac{1}{\epsilon})$ iteration for our algorithm⁷, and with possibly much larger constants in theory. Moreover, each iteration of their algorithm consists in solving a QP of increasing size (though they can use warm start to mitigate the repetition), hence it is subject to the same memory limitations as the QP formulation (2.15), and is not suitable for large-scale learning on hundreds of thousands of examples. Unfortunately, we are not aware of any empirical comparison between SVMstruct and either the dual extragradient or the QP formulation (2.15), hence it is not yet clear to what extent these conceptual considerations lead to significant differences in practice.

2.7.1 Online methods

Another approach to discriminative learning for structured prediction problems consists of online learning methods. We recall in passing that we used a “poor’s man regularization approach” in our experiments: rather than optimizing fully the large-margin objective for different values of the regularization parameters γ , we initialized the parameter at 0 and let our algorithm follow a path in the parameter space that we found empirically to stay somewhat close to the true regularization path. In the

⁷Note that both ϵ have slightly different meanings for the two algorithms: for SVMstruct, ϵ measures the maximum violation in the constraints, whereas our ϵ bounds the gap function (2.18).

context of large-scale learning, [Bottou and Bousquet \(2008\)](#) gave a compelling argument to motivate this kind of approach: for practitioners who are not interested in having an *interpretable model*, the goal is simply to minimize the generalization error in minimal computation time. This time can either be allocated to running the algorithm on more data (using a simple stochastic gradient technique for example), or to reducing the optimization error on the given data (which doesn't necessarily reduce the statistical error). First-order optimization methods seem a good compromise between the two goals. In our case, we also approximate the regularization path, though focusing on the generalization error rather than the exact optimization along the path.

An example of a successful online learning method for structured prediction is an extension of the MIRA algorithm of [Crammer and Singer \(2003\)](#) for structured prediction ([Crammer et al., 2005](#)), which was applied for example to dependency parsing ([McDonald et al., 2005](#)) and gene function prediction ([Bernal et al., 2007](#)). They obtained similar results as the QP (2.15) (which is equivalent to the max-margin Markov networks approach of [Taskar et al. \(2004b\)](#) for triangulated graphs) on the smaller dataset for which the QP could be run. They also provide regret bounds analyzing the convergence of a related version of their algorithm (see e.g. [Crammer et al., 2006](#)), though the practical implementation of their algorithm uses an approximate large-margin constraint only including the current k -best hypothesis according to the current score, rather than the full exponential set.

Recently, [Ratliff et al. \(2007\)](#) proposed to use the simple subgradient algorithm for structured prediction, which could be run either in batch mode or in online mode. Their algorithm consists of doing simple subgradient steps on the regularized structured hinge-loss (equation (2.21) with the ℓ_2 -norm of the parameter added). The subgradient method is a generalization of gradient descent for non-differentiable convex objectives, and has convergence rate of $\mathcal{O}(\frac{1}{\epsilon^2})$ steps for diminishing step sizes

under suitable regularity conditions – which are satisfied by the structured hinge loss (see e.g. [Boyd and Mutapcic, 2007](#), for a simple proof). [Ratliff et al. \(2007\)](#) also mentioned that the subgradient method with constant step size has *linear* convergence⁸, but the rate of convergence is only to a (possibly large) ball around the true solution, hence the rate here seems less meaningful. A subgradient of the hinge loss (2.21) (for one example i) is simply given by a difference of feature vectors $\mathbf{f}_i(\mathbf{y}^*) - \mathbf{f}_i(\mathbf{y}_i)$, where \mathbf{y}^* is a solution to the loss-augmented inference problem. The online update for the subgradient algorithm is thus quite similar to the perceptron algorithm, except it is using a solution to the loss-augmented inference problem rather than basic inference to compute the update direction as in the perceptron algorithm, and it contains as well a regularization term. This small difference is enough to enable [Ratliff et al. \(2007\)](#) to get better regret bounds and convergence guarantees than for the averaged perceptron. Moreover, they also provide some theoretical analysis for the case where approximate inference is used. Hence this approach seems to combine the generality advantage of SVMstruct (which can use more general loss functions) with the scalability advantage of the dual extragradient approach. Note that they have a slower convergence rate of $\mathcal{O}(\frac{1}{\epsilon^2})$, though. Interestingly, [Shalev-Shwartz et al. \(2007\)](#) were able to prove that their subgradient-based algorithm called PEGASOS had $\mathcal{O}(\frac{1}{\epsilon})$ rate, using elegant proof techniques from [Hazan et al. \(2007\)](#). However, they only derived this result for the binary SVM case, and it is unclear whether they could extend this result to the structured prediction case. Finally, we note that we can take advantage of a wider range of constraints on \mathcal{W} efficiently by using the non-Euclidean version of our algorithm. In particular, it can yield smaller Lipschitz constants and divergence radii and thus faster convergence in theory.

⁸Linear convergence applies to the case of strongly convex functions – such as ℓ_2 -norm regularized convex losses e.g.

2.8 Summary

We have presented a general and simple solution strategy for large-scale structured prediction problems. Using a saddle-point formulation of the problem, we exploit the dual extragradient algorithm, a simple gradient-based algorithm for saddle-point problems (Nesterov, 2003). The factoring of the problem into optimization over the feasible parameter space \mathcal{W} and feasible structured output space \mathcal{Z} allows easy integration of complex parameter constraints that arise in estimation of restricted classes of Markov networks and other models.

Key to our approach is the recognition that the projection step in the extragradient algorithm can be solved by network flow algorithms for matchings and min-cuts (and dynamic programming for decomposable models). Network flow algorithms are among the most well-developed algorithms in the field of combinatorial optimization, and yield stable, efficient algorithmic platforms.

One of the key bottlenecks of large learning problems is the memory requirement of the algorithm. We have derived a version of the algorithm that only uses storage proportional to the number of parameters in the model, and is independent of the number of examples. We have exhibited the favorable scaling of this overall approach in two concrete, large-scale learning problems. It is also important to note that the general approach extends and adopts to a much broader class of problems by allowing the use of Bregman projections suitable to particular problem classes.

Appendix 2.A Min-cut polytope projections

Consider projection for a single example i :

$$\begin{aligned} \min_{\mathbf{z}} \quad & \sum_{j \in \mathcal{V}} \frac{1}{2} (z'_j - z_j)^2 + \sum_{jk \in \mathcal{E}} \frac{1}{2} (z'_{jk} - z_{jk})^2 \\ \text{s.t.} \quad & 0 \leq z_j \leq 1, \quad \forall j \in \mathcal{V}; \quad z_j - z_k \leq z_{jk}, \quad z_k - z_j \leq z_{jk}, \quad \forall jk \in \mathcal{E}. \end{aligned} \quad (2.22)$$

Let $h_j^+(z_j) = \frac{1}{2}(z'_j - z_j)^2$ if $0 \leq z_j$, else ∞ . We introduce non-negative Lagrangian variables $\lambda_{jk}, \lambda_{kj}$ for the two constraints for each edge jk and λ_{j0} for the constraint $z_j \leq 1$ each node j .

The Lagrangian is given by:

$$\begin{aligned} L(\mathbf{z}, \lambda) = & \sum_{j \in \mathcal{V}} h_j^+(z_j) + \sum_{jk \in \mathcal{E}} \frac{1}{2} (z'_{jk} - z_{jk})^2 - \sum_{j \in \mathcal{V}} (1 - z_j) \lambda_{j0} \\ & - \sum_{jk \in \mathcal{E}} (z_{jk} - z_j + z_k) \lambda_{jk} - \sum_{jk \in \mathcal{E}} (z_{jk} - z_k + z_j) \lambda_{kj} \end{aligned}$$

Letting $\lambda_{0j} = \lambda_{j0} + \sum_{k:jk \in \mathcal{E}} (\lambda_{jk} - \lambda_{kj}) + \sum_{k:kj \in \mathcal{E}} (\lambda_{jk} - \lambda_{kj})$, note that

$$\sum_{j \in \mathcal{V}} z_j \lambda_{0j} = \sum_{j \in \mathcal{V}} z_j \lambda_{j0} + \sum_{jk \in \mathcal{E}} (z_j - z_k) \lambda_{jk} + \sum_{jk \in \mathcal{E}} (z_k - z_j) \lambda_{kj}.$$

So the Lagrangian becomes:

$$L(\mathbf{z}, \lambda) = \sum_{j \in \mathcal{V}} [h_j^+(z_j) + z_j \lambda_{0j} - \lambda_{j0}] + \sum_{jk \in \mathcal{E}} \left[\frac{1}{2} (z'_{jk} - z_{jk})^2 - z_{jk} (\lambda_{jk} + \lambda_{kj}) \right].$$

Now, minimizing $L(\mathbf{z}, \lambda)$ with respect to \mathbf{z} , we have

$$\min_{\mathbf{z}} L(\mathbf{z}, \lambda) = \sum_{jk \in \mathcal{E}} q_{jk} (\lambda_{jk} + \lambda_{kj}) + \sum_{j \in \mathcal{V}} [q_{0j} (\lambda_{0j}) - \lambda_{j0}],$$

where

$$\begin{aligned} q_{jk}(\lambda_{jk} + \lambda_{kj}) &= \min_{z_{jk}} \left[\frac{1}{2}(z'_{jk} - z_{jk})^2 - z_{jk}(\lambda_{jk} + \lambda_{kj}) \right] \\ q_{0j}(\lambda_{0j}) &= \min_{z_j} [h_j^+(z_j) + z_j \lambda_{0j}]. \end{aligned}$$

The minimizing values of \mathbf{z} are:

$$\begin{aligned} z_j^* &= \arg \min_{z_j} [h_j^+(z_j) + z_j \lambda_{0j}] = \begin{cases} 0 & \lambda_{0j} \geq z'_j; \\ z'_j - \lambda_{0j} & \lambda_{0j} \leq z'_j; \end{cases} \\ z_{jk}^* &= \arg \min_{z_{jk}} \left[\frac{1}{2}(z'_{jk} - z_{jk})^2 - z_{jk}(\lambda_{jk} + \lambda_{kj}) \right] = z'_{jk} + \lambda_{jk} + \lambda_{kj}. \end{aligned}$$

Hence, we have:

$$\begin{aligned} q_{jk}(\lambda_{jk} + \lambda_{kj}) &= -z'_{jk}(\lambda_{jk} + \lambda_{kj}) - \frac{1}{2}(\lambda_{jk} + \lambda_{kj})^2 \\ q_{0j}(\lambda_{0j}) &= \begin{cases} \frac{1}{2}z_j'^2 & \lambda_{0j} \geq z'_j; \\ z'_j \lambda_{0j} - \frac{1}{2}\lambda_{0j}^2 & \lambda_{0j} \leq z'_j. \end{cases} \end{aligned}$$

The dual of the projection problem is thus:

$$\begin{aligned} \max_{\lambda} \quad & \sum_{j \in \mathcal{V}} [q_{0j}(\lambda_{0j}) - \lambda_{j0}] + \sum_{jk \in \mathcal{E}} \left[-z'_{jk}(\lambda_{jk} + \lambda_{kj}) - \frac{1}{2}(\lambda_{jk} + \lambda_{kj})^2 \right] \quad (2.23) \\ \text{s.t.} \quad & \lambda_{j0} - \lambda_{0j} + \sum_{jk \in \mathcal{E}} (\lambda_{jk} - \lambda_{kj}) = 0, \quad \forall j \in \mathcal{V}; \\ & \lambda_{jk}, \lambda_{kj} \geq 0, \quad \forall jk \in \mathcal{E}; \quad \lambda_{j0} \geq 0, \quad \forall j \in \mathcal{V}. \end{aligned}$$

Interpreting λ_{jk} as flow from node j to node k , and λ_{kj} as flow from k to j and $\lambda_{j0}, \lambda_{0j}$ as flow from and to a special node 0, we can identify the constraints of Eq. (2.23) as conservation of flow constraints. The last transformation we need is to address the presence of cross-terms $\lambda_{jk}\lambda_{kj}$ in the objective. Note that in the flow conservation

constraints, $\lambda_{jk}, \lambda_{kj}$ always appear together as $\lambda_{jk} - \lambda_{kj}$. Since we are minimizing $(\lambda_{jk} + \lambda_{kj})^2$ subject to constraints on $\lambda_{jk} - \lambda_{kj}$, at least one of $\lambda_{jk}, \lambda_{kj}$ will be zero at the optimum and the cross-terms can be ignored. Note that all λ variables are non-negative except for λ_{0j} 's. Many standard flow packages support this problem form, but we can also transform the problem to have all non-negative flows by introducing extra variables. The final form has a convex quadratic cost for each edge:

$$\begin{aligned}
 \min_{\lambda} \quad & \sum_{j \in \mathcal{V}} [-q_{0j}(\lambda_{0j}) + \lambda_{j0}] + \sum_{jk \in \mathcal{E}} \left[z'_{jk} \lambda_{jk} + \frac{1}{2} \lambda_{jk}^2 \right] + \sum_{jk \in \mathcal{E}} \left[z'_{jk} \lambda_{kj} + \frac{1}{2} \lambda_{kj}^2 \right] \quad (2.24) \\
 \text{s.t.} \quad & \lambda_{j0} - \lambda_{0j} + \sum_{jk \in \mathcal{E}} (\lambda_{jk} - \lambda_{kj}) = 0, \quad \forall j \in \mathcal{V}; \\
 & \lambda_{jk}, \lambda_{kj} \geq 0, \quad \forall jk \in \mathcal{E}; \quad \lambda_{j0} \geq 0, \quad \forall j \in \mathcal{V}.
 \end{aligned}$$

Chapter 3

Word Alignment via Quadratic Assignment

In the previous chapter, we presented a general formulation for large margin learning of structured output models for which we also had an efficient optimization algorithm. An example of application was the word alignment problem from machine translation. In this chapter, we focus on this application and show how we can obtain state-of-the-art results by designing our model more carefully, while still staying inside the large margin learning framework for structured prediction.

3.1 Background

Word alignment is a key component of most end-to-end statistical machine translation systems. The standard approach to word alignment is to construct directional generative models (Brown et al., 1990; Och and Ney, 2003), which produce a sentence in one language given the sentence in another language. While these models require sentence-aligned bitexts, they can be trained with no further supervision, using EM. Generative alignment models do, however, have serious drawbacks. First,

they require extensive tuning and processing of large amounts of data which, for the better-performing models, is a non-trivial resource requirement. Second, conditioning on arbitrary features of the input is difficult; for example, we would like to condition on the orthographic similarity of a word pair (for detecting cognates), the presence of that pair in various dictionaries, the similarity of the frequency of its two words, choices made by other alignment systems, and so on.

Recently, [Moore \(2005\)](#) proposed a discriminative model in which pairs of sentences (e, f) and proposed alignments a are scored using a linear combination of arbitrary features computed from the tuples (a, e, f) . While there are no restrictions on the form of the model features, the problem of finding the highest scoring alignment is very difficult and involves heuristic search. Moreover, the parameters of the model must be estimated using averaged perceptron training ([Collins, 2002](#)), which can be unstable. In contrast, in the approach we presented in [Sec. 2.2.3](#) (and also described in [Taskar et al. \(2005b\)](#)), we cast word alignment as a maximum weighted matching problem, in which each pair of words (e_j, f_k) in a sentence pair (e, f) is associated with a score $s_{jk}(e, f)$ reflecting the desirability of the alignment of that pair. Importantly, this approach is computationally tractable. The alignment for the sentence pair is the highest scoring matching under constraints (such as the constraint that matchings be one-to-one). The scoring model $s_{jk}(e, f)$ can be based on a rich feature set defined on word pairs (e_j, f_k) and their context, including measures of association, orthography, relative position, predictions of generative models, etc. The parameters of the model are estimated within the framework of large-margin estimation; in particular, the problem turns out to reduce to the solution of a (relatively) small quadratic program (QP) or can be solved as a saddle-point formulation which can be scaled to large datasets. We also showed that large-margin estimation is both more stable and more accurate than perceptron training.

While the bipartite matching approach is a useful first step in the direction of

discriminative word alignment, for discriminative approaches to compete with and eventually surpass the most sophisticated generative models, it is necessary to consider more realistic underlying statistical models. Note in particular two substantial limitations of the bipartite matching model that we presented in Sec. 2.2.3: words have fertility of at most one, and there is no way to incorporate pairwise interactions among alignment decisions. Moving beyond these limitations—while retaining computational tractability—is the next major challenge for discriminative word alignment.

In this chapter, we show how to overcome both limitations. First, we introduce a parameterized model that penalizes different levels of fertility. While this extension adds very useful expressive power to the model, it turns out not to increase the computational complexity of the aligner, for either the prediction or the parameter estimation problem. Second, we introduce a more thoroughgoing extension which incorporates first-order interactions between alignments of consecutive words into the model. We do this by formulating the alignment problem as a quadratic assignment problem (QAP), where in addition to scoring individual edges, we also define scores of pairs of edges that connect consecutive words in an alignment. The predicted alignment is the highest scoring quadratic assignment.

QAP is an NP-hard problem, but in the range of problem sizes that we need to tackle, the problem can be solved efficiently. In particular, using standard off-the-shelf integer program solvers, we are able to solve the QAP problems in our experiments in under a second. Moreover, the parameter estimation problem can also be solved efficiently by making use of a linear relaxation of QAP for the min-max formulation of large-margin estimation (Taskar, 2004).

We show that these two extensions yield significant improvements in error rates when compared to the bipartite matching model. The addition of a fertility model

improves the AER¹ by 0.4. Modeling first-order interactions improves the AER by 1.8. Combining the two extensions results in an improvement in AER of 2.3, yielding alignments of better quality than intersected IBM Model 4. Moreover, including predictions of bi-directional IBM Model 4 and model of [Liang et al. \(2006\)](#) as features, we achieve an absolute AER of 3.8 on the English-French Hansards alignment task—the best AER result published on this task to date.

3.2 Models

Recall the maximum weight bipartite matching model for word alignment of [Sec. 2.2.3](#). Nodes $\mathcal{V} = \mathcal{V}^s \cup \mathcal{V}^t$ correspond to words in the “source” (\mathcal{V}^s) and “target” (\mathcal{V}^t) sentences, and edges $\mathcal{E} = \{jk : j \in \mathcal{V}^s, k \in \mathcal{V}^t\}$ correspond to alignments between word pairs.² The edge weights s_{jk} represent the degree to which word j in one sentence can be translated using the word k in the other sentence. The predicted alignment is chosen by maximizing the sum of edge scores. A matching is represented using a set of binary variables y_{jk} that are set to 1 if word j is assigned to word k in the other sentence, and 0 otherwise. The score of an assignment is the sum of edge scores: $s(\mathbf{y}) = \sum_{jk} s_{jk}y_{jk}$. In our base model, we had assumed that each word aligns to one or zero words in the other sentence; we revisit the issue of fertility in the next section. Recall that the maximum weight bipartite matching problem, $\arg \max_{\mathbf{y} \in \mathcal{Y}} s(\mathbf{y})$, can be solved using combinatorial algorithms for min-cost max-flow, expressed in a linear

¹AER was defined in [Sec. 2.6.2](#) and will be expressed in percent in this chapter for convenience.

²The source/target designation is arbitrary, as the models considered below are all symmetric.

programming (LP) formulation as follows:

$$\begin{aligned} \max_{0 \leq \mathbf{z} \leq 1} \quad & \sum_{jk \in \mathcal{E}} s_{jk} z_{jk} \\ \text{s.t.} \quad & \sum_{j \in \mathcal{V}^s} z_{jk} \leq 1, \forall k \in \mathcal{V}^t; \quad \sum_{k \in \mathcal{V}^t} z_{jk} \leq 1, \forall j \in \mathcal{V}^s, \end{aligned} \tag{3.1}$$

where the continuous variables z_{jk} are a relaxation of the corresponding binary-valued variables y_{jk} . This LP is guaranteed to have integral (and hence optimal) solutions for any scoring function $s(\mathbf{y})$ (Schrijver, 2003). Note that although the above LP can be used to compute alignments, combinatorial algorithms are generally more efficient. For example, in Fig. 3.1(a), we show a standard construction for an equivalent min-cost flow problem. However, we build on this LP to develop our extensions to this model below. Representing the prediction problem as an LP or an integer LP provides a precise (and concise) way of specifying the model and allows us to use the large-margin framework of Taskar (2004) that we reviewed in Sec. 2.3. As before, we let $s_{jk} = \mathbf{w}^\top \mathbf{f}(\mathbf{x}_{jk})$ for a feature mapping \mathbf{f} which can include the identity of the two words, their relative positions in their respective sentences, their part-of-speech tags, their string similarity (for detecting cognates), and so on.

3.2.1 Fertility

An important limitation of the model in Eq. (3.1) is that in each sentence, a word can align to at most one word in the translation. Although it is common that words have gold fertility zero or one, it is certainly not always true. Consider, for example, the bitext fragment shown in Fig. 3.2(a), where *backbone* is aligned to the phrase *épine dorsale*. In this figure, outlines are gold alignments, square for sure alignments, round for possibles, and filled squares are target alignments (for details on gold alignments, see Sec. 3.4). When considering only the sure alignments on the standard Hansards

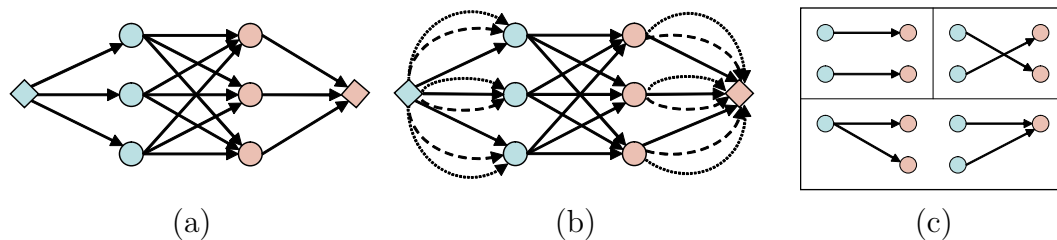


Figure 3.1: (a) Maximum weight bipartite matching as min-cost flow. Diamond-shaped nodes represent flow source and sink. All edge capacities are 1, with edges between round nodes (j, k) have cost $-s_{jk}$, edges from source and to sink have cost 0. (b) Expanded min-cost flow graph with new edges from source and to sink that allow fertility of up to 3. The capacities of the new edges are 1 and the costs are 0 for solid edges from source and to sink, $s_{2j\bullet}$, $s_{2\bullet k}$ for dashed edges, and $s_{3j\bullet}$, $s_{3\bullet k}$ for dotted edges. (c) Three types of pairs of edges included in the QAP model, where the nodes on both sides correspond to consecutive words.

dataset, 7 percent of the word occurrences have fertility 2, and 1 percent have fertility 3 and above; when considering the possible alignments high fertility is much more common—31 percent of the words have fertility 3 and above.

One simple fix to the original matching model is to increase the right hand sides for the constraints in Eq. (3.1) from 1 to D , where D is the maximum allowed fertility. However, this change results in an undesirable bimodal behavior, where maximum weight solutions either have all words with fertility 0 or D , depending on whether most scores s_{jk} are positive or negative. For example, if scores tend to be positive, most words will want to collect as many alignments as they are permitted. What the model is missing is a means for encouraging the common case of low fertility (0 or 1), while allowing higher fertility when it is licensed. This end can be achieved by introducing a penalty for having higher fertility, with the goal of allowing that penalty to vary based on features of the word in question (such as its frequency or identity).

In order to model such a penalty, we introduce indicator variables $z_{dj\bullet}$ (and $z_{d\bullet k}$)

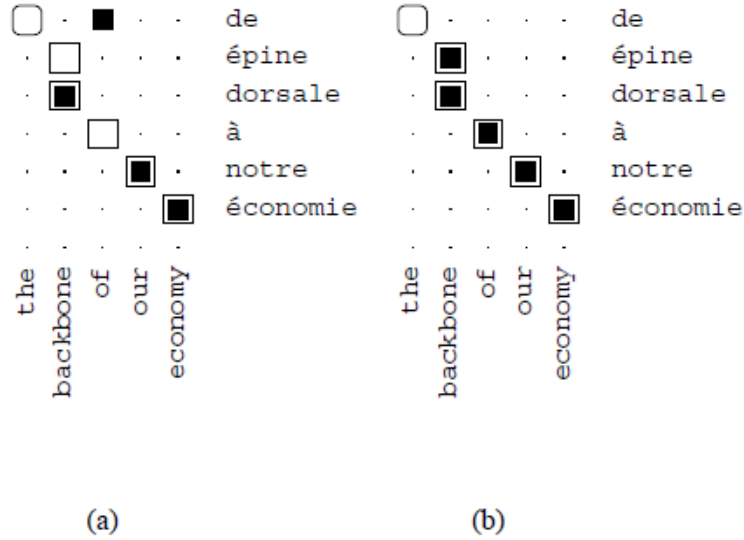


Figure 3.2: An example fragment that requires fertility greater than one to correctly label. (a) The guess of the baseline M model. (b) The guess of the M+F fertility-augmented model.

with the intended meaning: node j has fertility of at least d (and node k has fertility of at least d). In the following LP, we introduce a penalty of $\sum_{2 \leq d \leq D} s_{dj\bullet} z_{dj\bullet}$ for fertility of node j , where each term $s_{dj\bullet} \geq 0$ is the penalty increment for increasing the fertility from $d - 1$ to d :

$$\begin{aligned}
 \max_{0 \leq z \leq 1} \quad & \sum_{jk \in \mathcal{E}} s_{jk} z_{jk} - \sum_{j \in \mathcal{V}^s, 2 \leq d \leq D} s_{dj\bullet} z_{dj\bullet} - \sum_{k \in \mathcal{V}^t, 2 \leq d \leq D} s_{d\bullet k} z_{d\bullet k} \quad (3.2) \\
 \text{s.t.} \quad & \sum_{j \in \mathcal{V}^s} z_{jk} \leq 1 + \sum_{2 \leq d \leq D} z_{d\bullet k}, \quad \forall k \in \mathcal{V}^t; \\
 & \sum_{k \in \mathcal{V}^t} z_{jk} \leq 1 + \sum_{2 \leq d \leq D} z_{dj\bullet}, \quad \forall j \in \mathcal{V}^s.
 \end{aligned}$$

We can show that this LP always has integral solutions by a reduction to a min-cost flow problem. The construction is shown in Fig. 3.1(b). To ensure that the new

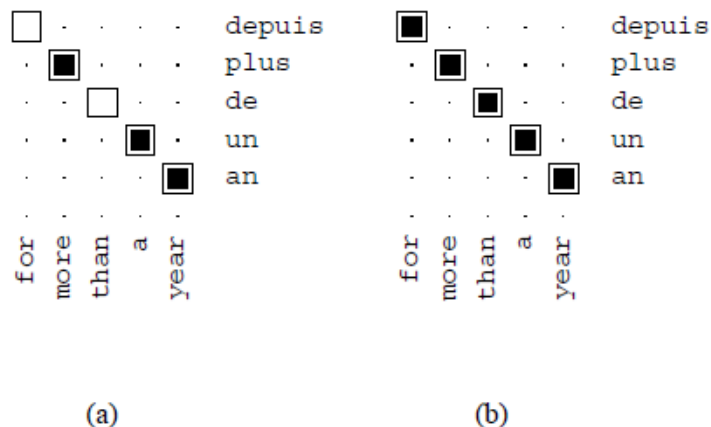


Figure 3.3: An example fragment with a monotonic gold alignment. (a) The guess of the baseline M model. (b) The guess of the M+Q quadratic model.

variables have the intended semantics, we need to make sure that $s_{dj\bullet} \leq s_{d'j\bullet}$ if $d \leq d'$, so that the lower cost $z_{dj\bullet}$ is used before the higher cost $z_{d'j\bullet}$ to increase fertility. This restriction implies that the penalty must be monotonic and convex as a function of the fertility.

To anticipate the results that we report in Sec. 3.4, adding fertility to the basic matching model makes the target alignment of the *backbone* example feasible and, in this case, the model correctly labels this fragment as shown in Fig. 3.2(b).

3.2.2 First-order interactions

An even more significant limitation of the model in Eq. (3.1) is that the edges interact only indirectly through the competition induced by the constraints. Generative alignment models like the HMM model (Vogel et al., 1996) and IBM models 4 and above (Brown et al., 1990; Och and Ney, 2003) directly model correlations between

alignments of consecutive words (at least on one side). For example, Fig. 3.3 shows a bitext fragment whose gold alignment is strictly monotonic. This monotonicity is quite common – 46% of the words in the hand-aligned data diagonally follow a previous alignment in this way. We can model the common local alignment configurations by adding bonuses for pairs of edges. For example, strictly monotonic alignments can be encouraged by boosting the scores of edges of the form $\langle (j, k), (j + 1, k + 1) \rangle$. Another trend, common in English-French translation (7% on the hand-aligned data), is the local inversion of nouns and adjectives, which typically involves a pair of edges $\langle (j, k + 1), (j + 1, k) \rangle$. Finally, a word in one language is often translated as a phrase (consecutive sequence of words) in the other language. This pattern involves pairs of edges with the same origin on one side: $\langle (j, k), (j, k + 1) \rangle$ or $\langle (j, k), (j + 1, k) \rangle$. All three of these edge pair patterns are shown in Fig. 3.1(c). Note that the set of such edge pairs $\mathcal{Q} = \{jklm : |j - l| \leq 1, |k - m| \leq 1\}$ is of linear size in the number of edges.

Formally, we add to the model variables z_{jklm} which indicate whether both edge jk and lm are in the alignment. We also add a corresponding score s_{jklm} , which we assume to be non-negative, since the correlations we described are positive. (Negative scores can also be used, but the resulting formulation we present below would be slightly different.) To enforce the semantics $z_{jklm} = z_{jk}z_{lm}$, we use a pair of constraints $z_{jklm} \leq z_{jk}; z_{jklm} \leq z_{lm}$. Since s_{jklm} is positive, at the optimum, $z_{jklm} = \min(z_{jk}, z_{lm})$. If in addition z_{jk}, z_{lm} are integral (0 or 1), then $z_{jklm} = z_{jk}z_{lm}$. Hence, solving the following mixed integer program will find the optimal quadratic assignment for our

model (called the QAP model hereafter):

$$\begin{aligned}
 \max_{\mathbf{z} \in \{0,1\}} \quad & \sum_{jk \in \mathcal{E}} s_{jk} z_{jk} + \sum_{jklm \in \mathcal{Q}} s_{jklm} z_{jklm} & (3.3) \\
 \text{s.t.} \quad & \sum_{j \in \mathcal{V}^s} z_{jk} \leq 1, \quad \forall k \in \mathcal{V}^t; \quad \sum_{k \in \mathcal{V}^t} z_{jk} \leq 1, \quad \forall j \in \mathcal{V}^s; \\
 & z_{jklm} \leq z_{jk}, \quad z_{jklm} \leq z_{lm}, \quad \forall jklm \in \mathcal{Q}.
 \end{aligned}$$

Note that we can also combine this extension with the fertility extension described above.

To once again anticipate the results presented in Sec. 3.4, our baseline model of Chapter 2 makes the prediction given in Fig. 3.3(a) because the two missing alignments are atypical translations of common words. With the addition of edge pair features, the overall monotonicity pushes the alignment to that of Fig. 3.3(b).

3.3 Parameter estimation

To estimate the parameters of our model, we follow the large-margin formulation of Sec. 2.3. Our input is a set of training instances $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$, where each instance consists of a sentence pair \mathbf{x}_i and a target alignment \mathbf{y}_i . We would like to find parameters \mathbf{w} that predict correct alignments on the training data: $\mathbf{y}_i = \arg \max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \mathbf{y}'_i)$ for each i , where \mathcal{Y}_i is the space of matchings for the sentence pair \mathbf{x}_i with appropriate fertility constraints depending on which model we are using.

In standard classification problems, we typically measure the error of prediction, $\ell(\mathbf{y}_i, \mathbf{y}'_i)$, using the simple 0-1 loss. In structured problems, where we are jointly predicting multiple variables, the loss is often more complex. While the F-measure is a natural loss function for this task, we instead chose a sensible surrogate that fits better in our framework: weighted Hamming distance, which counts the number

of variables in which a candidate solution \mathbf{y}' differs from the target output \mathbf{y} , with different penalty for false positives (c^+) and false negatives (c^-):

$$\ell(\mathbf{y}, \mathbf{y}') = \sum_{jk} [c^+(1 - y_{jk})y'_{jk} + c^-(1 - y'_{jk})y_{jk}].$$

Following the notation of Sec. 2.3, we estimate the parameters \mathbf{w} by minimizing an SVM-like hinge upper bound on the total structured loss $\sum_i \ell(\mathbf{y}_i, h_{\mathbf{w}}(\mathbf{x}_i))$:

$$\min_{\|\mathbf{w}\| \leq \gamma} \sum_i \max_{\mathbf{y}'_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i)] - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i), \quad (3.4)$$

where γ is a regularization parameter.

In this form, the estimation problem is a mixture of continuous optimization over \mathbf{w} and combinatorial optimization over \mathbf{y}_i . In order to transform it into a more standard optimization problem, we need a way to efficiently handle the *loss-augmented inference*, $\max_{\mathbf{y}'_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i)]$. This optimization problem has precisely the same form as the prediction problem whose parameters we are trying to learn — $\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i)$ — but with an additional term corresponding to the loss function. Our assumption that the loss function decomposes over the edges is crucial to solving this problem. As in Sec. 2.3, the prediction problems (3.1) and (3.2) as well as the relaxed form of (3.3) are LPs and hence we can express the loss-augmented inference as an LP for each example i :

$$\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i) = d_i + \max_{\mathbf{z}_i \in \mathcal{Z}_i} (\mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i)^\top \mathbf{z}_i, \quad (3.5)$$

for appropriately defined d_i , \mathbf{c}_i , \mathbf{F}_i and \mathcal{Z}_i encoding the corresponding LP of Sec. 3.2. For example for the first order interaction model (3.3), \mathbf{z}_i is a stacked vector of the edge variables $z_{i,jk}$ for $jk \in \mathcal{E}_i$ and the pair of edges variables $z_{i,jklm}$ for $jklm \in \mathcal{Q}_i$;

\mathcal{Z}_i is the convex polytope defined by the linear constraints of (3.3) (excluding the integer constraints); \mathbf{F}_i is a matrix whose first columns are the features $\mathbf{f}(\mathbf{x}_{i,jk})$ for each edge in example i and last columns are the features $\mathbf{f}(\mathbf{x}_{i,jklm})$ for each pair of edges in \mathcal{Q}_i ; $d_i = \sum_{jk} c^- y_{i,jk}$ and \mathbf{c}_i is the vector of the loss terms $c^+ - (c^- + c^+)y_{i,jk}$, padded with zeros for the $jklm$ components. Plugging (3.5) back into the estimation equation (3.4) and using the notation of Sec. 2.3 yields the saddle-point formulation:

$$\min_{\|\mathbf{w}\| \leq \gamma} \max_{\mathbf{z} \in \mathcal{Z}} \sum_i (\mathbf{w}^\top \mathbf{F}_i \mathbf{z}_i + \mathbf{c}_i^\top \mathbf{z}_i - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i)) \quad (3.6)$$

where we have omitted the constant term $\sum_i d_i$. This formulation is equivalent to (3.4) for the basic and fertility models, and for the QAP model (3.3) if we add an integer constraint in the definition of \mathcal{Z}_i . For the relaxed \mathcal{Z}_i without the integer constraint, it corresponds to minimizing an upper bound on (3.4), as each LP gives an upper bound on the corresponding integer program.

At this point, we could proceed to use the dual extragradient optimization method of Sec. 2.4 which enables us to train our model on hundreds of thousands of sentences. This is possible for both the basic model (3.1) and the fertility model (3.2) as their LP can be represented as a min-cost network flow problem, enabling us to use the efficient projection methods presented in Sec. 2.4.1.2. On the other hand, we don't know how to reduce the QAP formulation (3.3) to a flow formulation, hence a standard QP solver would be needed for the (relaxed) form of the projection, or a quadratic integer solver for the exact formulation. Modern off-the-shelf solvers such as MOSEK or CPLEX can solve these efficiently for small problem sizes, even though the QAP problem is NP-complete. For simplicity of comparison of different models though, we opted to use the Quadratically Constrained Quadratic Program (QCQP) formulation given in equation (2.15), obtained by dualizing the LPs in (3.6). This formulation is

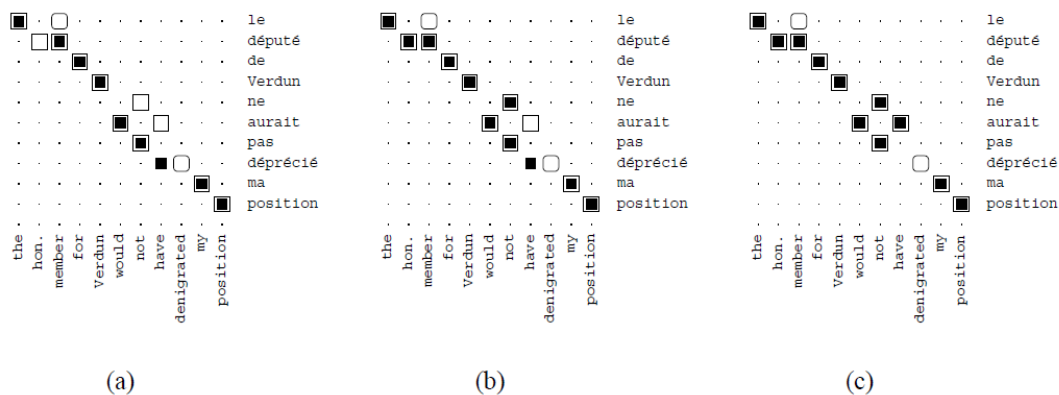


Figure 3.4: An example fragment with several multiple fertility sure alignments. (a) The guess of the M+Q model with maximum fertility of one. (b) The guess of the M+Q+F quadratic model with fertility two permitted. (c) The guess of the M+Q+F model with lexical fertility features.

exact for the basic and fertility model, and corresponds to minimizing an upper bound on the true hinge loss for the QAP model. Using the LP relaxation for the large-margin QCQP formulation is an approximation, but as our experiments indicate, this approximation is very effective. At testing time, though, we use the integer LP to predict alignments³.

3.4 Experiments

We applied our algorithms to word-level alignment using the English-French Hansards data described in Sec. 2.2.3. We also use the *alignment error rate* (AER) as our evaluation metric. Fig. 3.4 illustrates the annotation for the dataset: proposed alignments are shown against gold alignments, with open squares for sure alignments, rounded

³When training on 200 sentences, the QCQP we obtain contains roughly 700K variables and 300K constraints and is solved in roughly 10 minutes on a 2.8 GHz Pentium 4 machine using MOSEK. Aligning the whole training set with the flow formulation takes a few seconds, whereas using the integer programming (for the QAP formulation) takes 1-2 minutes.

open squares for possible alignments, and filled black squares for proposed alignments.

The input to our algorithm is a small number of labeled examples. In order to make our results more comparable with Moore (2005), we split the original set into 200 training examples and 247 test examples. We also trained on only the first 100 to make our results more comparable with the experiments of Och and Ney (2003), in which IBM model 4 was tuned using 100 sentences. In all our experiments, we used a structured loss function that penalized false negatives 10 times more than false positives, where the value of 10 was picked by using a validation set. The regularization parameter γ was also chosen using the validation set. Our code is available from <http://www.cs.berkeley.edu/~slacoste/research/qap/>.

3.4.1 Features and results

We parameterized all scoring functions s_{jk} , $s_{dj\bullet}$, $s_{d\bullet k}$ and s_{jklm} as weighted linear combinations of feature sets. The features were computed from the large unlabeled corpus of 1.1M automatically aligned sentences.

In the remainder of this section we describe the improvements to the model performance as various features are added. One of the most useful features for the basic matching model is, of course, the set of predictions of IBM model 4. However, computing these features is very expensive and we would like to build a competitive model that doesn't require them. Instead, we made significant use of IBM model 2 as a source of features. This model, although not very accurate as a predictive model, is simple and cheap to construct and it is a useful source of features.

3.4.1.1 The Basic Matching Model: Edge Features

In the basic matching model of Sec. 2.2.3, called M here, one can only specify features on pairs of word tokens, i.e. alignment edges. These features include word association,

orthography, proximity, etc., and are documented in more details in Taskar et al. (2005b). We also augment those features with the predictions of IBM Model 2 run on the training and test sentences. We provided features for model 2 trained in each direction, as well as the intersected predictions, on each edge. By including the IBM Model 2 features, the performance of the model described in Taskar et al. (2005b) on our test set (trained on 200 sentences) improves from 10.0 AER to 8.2 AER, outperforming unsymmetrized IBM Model 4 (but not intersected model 4).

As an example of the kinds of errors the baseline M system makes, see Fig. 3.2 (where multiple fertility cannot be predicted), Fig. 3.3 (where a preference for monotonicity cannot be modeled), and Fig. 3.4 (which shows several multi-fertile cases).

3.4.1.2 The Fertility Model: Node Features

To address errors like those shown in Fig. 3.2, we increased the maximum fertility to two using the parameterized fertility model of Sec. 3.2.1. The model learns costs on the second flow arc for each word via features not of edges but of single words. The score of taking a second match for a word w was based on the following features: a bias feature, the proportion of times w 's type was aligned to two or more words by IBM model 2, and the bucketed frequency of the word type. This model was called M+F. We also included a lexicalized feature for words which were common in our training set: whether w was ever seen in a multiple fertility alignment (more on this feature later). This enabled the system to learn that certain words, such as the English *not* and French verbs like *aurait* commonly participate in multiple fertility configurations.

Table 3.1 show the results using the fertility extension. Adding fertility lowered AER from 8.5 to 8.1, though fertility was even more effective in conjunction with the quadratic features below. The M+F setting was even able to correctly learn some multiple fertility instances which were not seen in the training data, such as those

shown in Fig. 3.2.

3.4.1.3 The First-Order Model: Quadratic Features

With or without the fertility model, the model makes mistakes such as those shown in Fig. 3.3, where atypical translations of common words are not chosen despite their local support from adjacent edges. In the quadratic model, we can associate features with pairs of edges. We began with features which identify each specific pattern, enabling trends of monotonicity (or inversion) to be captured. We also added to each edge pair the fraction of times that pair’s pattern (monotonic, inverted, one to two) occurred according each version of IBM model 2 (forward, backward, intersected).

Table 3.1 shows the results of adding the quadratic model. M+Q reduces error over M from 8.5 to 6.7 (and fixes the errors shown in Fig. 3.3). When both the fertility and quadratic extensions were added, AER dropped further, to 6.2. This final model is even able to capture the diamond pattern in Fig. 3.4; the adjacent cycle of alignments is reinforced by the quadratic features which boost adjacency. The example in Fig. 3.4 shows another interesting phenomenon: the multi-fertile alignments for *not* and *député* are learned even without lexical fertility features (Fig. 3.4b), because the Dice coefficients of those words with their two alignees are both high. However the surface association of *aurait* with *have* is much higher than with *would*. If, however, lexical features are added, *would* is correctly aligned as well (Fig. 3.4c), since it is observed in similar periphrastic constructions in the training set.

We have avoided using expensive-to-compute features like IBM model 4 predictions up to this point. However, if these are available, our model can improve further. By adding model 4 predictions to the edge features, we get a relative AER reduction of 27%, from 6.5 to 4.5. By also including as features the posteriors of the model of Liang et al. (2006), we achieve AER of 3.8, and 96.7/95.5 precision/recall.

It is comforting to note that in practice, the burden of running an integer linear

Model	Prec	Rec	AER
Generative			
IBM 2 (E→F)	73.6	87.7	21.7
IBM 2 (F→E)	75.4	87.0	20.6
IBM 2 (intersected)	90.1	80.4	14.3
IBM 4 (E→F)	90.3	92.1	9.0
IBM 4 (F→E)	90.8	91.3	9.0
IBM 4 (intersected)	98.0	88.1	6.5
Discriminative (100 sentences)			
Matching (M)	94.1	88.5	8.5
M + Fertility (F)	93.9	89.4	8.1
M + Quadratic (Q)	94.4	91.9	6.7
M + F + Q	94.8	92.5	6.2
M + F + Q + IBM4	96.4	94.4	4.5
M + F + Q + IBM4 + Liang	96.7	95.5	3.8
Discriminative (200 sentences)			
Matching (M)	93.4	89.7	8.2
M + Fertility (F)	93.6	90.1	8.0
M + Quadratic (Q)	95.0	91.1	6.8
M + F + Q	95.2	92.4	6.1
M + F + Q + IBM4	96.0	95.0	4.4

Table 3.1: AER results on the Hansards task.

program at test time can be avoided. We experimented with using just the LP relaxation and found that on the test set, only about 20% of sentences have fractional solutions and only 0.2% of all edges are fractional. Simple rounding of each edge value in the LP solution achieves the same AER as the integer LP solution, while using about a third of the computation time on average.

3.5 Summary

We have shown that the discriminative approach to word alignment can be extended to allow flexible fertility modeling and to capture first-order interactions between alignments of consecutive words. These extensions significantly enhance the expres-

sive power of the discriminative approach; in particular, they make it possible to capture phenomena of monotonicity, local inversion and contiguous fertility trends—phenomena that are highly informative for alignment. They do so while remaining computationally efficient in practice both for prediction and for parameter estimation.

Our best model achieves a relative AER reduction of 25% over the basic matching formulation, beating intersected IBM Model 4 without the use of any compute-intensive features. Including Model 4 predictions as features, we achieve a further relative AER reduction of 32% over intersected Model 4 alignments. By also including predictions of another model, we drive AER down to 3.8. An important question left for future work is whether the improvement in AER results in better translation BLEU score. Allowing higher fertility and optimizing a recall biased cost function provide a significant increase in recall relative to the intersected IBM model 4 (from 88.1% to 94.4%), with only a small degradation in precision. We view this as a particularly promising aspect of our work, given that phrase-based machine translation systems such as Pharaoh ([Koehn et al., 2003](#)) perform better with higher recall alignments.

Chapter 4

DiscLDA: Discriminative Dimensionality Reduction for Classification

In the two previous chapters, we saw how the combinatorial structure of structured prediction problems could be exploited to obtain efficient discriminative learning approaches with better predictive performance than the previous generative learning models. In this chapter, we focus our attention to a different type of structure that we would like to exploit with our discriminative learning toolbox: a latent-variable representation of the inputs. The question we have in mind is how one can combine the useful interpretability of a generative model on the inputs, while preserving discriminative power for a classification task. This question can be approached from several directions. In this thesis, we explore it from the point of view of discriminative dimensionality reduction.

4.1 Background

Dimensionality reduction is a common and often necessary step in many machine learning applications and high-dimensional data analyses. There is a rich history and literature on the subject, ranging from classical linear methods such as principal component analysis (PCA) and Fisher discriminant analysis (FDA) to a variety of nonlinear procedures such as kernelized versions of PCA and FDA as well as manifold learning algorithms.

A recent trend in dimensionality reduction is to focus on probabilistic models. These models, which include generative topological mapping, factor analysis, independent component analysis and probabilistic latent semantic analysis (pLSA), are generally specified in terms of an underlying independence assumption or low-rank assumption. The models are generally fit with maximum likelihood, although Bayesian methods are sometimes used. In particular, Latent Dirichlet Allocation (LDA) is a Bayesian model in the spirit of pLSA that models each data point (e.g., a document) as a collection of draws from a mixture model in which each mixture component is known as a *topic* (Blei et al., 2003). The mixing proportions across topics are document-specific, and the posterior distribution across these mixing proportions provides a reduced representation of the document. This model has been used successfully in a number of applied domains, including information retrieval, vision and bioinformatics (Griffiths and Steyvers, 2004; Berg et al., 2004).

These dimensionality reduction methods are entirely unsupervised. Another branch of research, known as *sufficient dimension reduction* (SDR), aims at making use of supervisory data in dimension reduction (Chiaromonte and Cook, 2002; Fukumizu et al., 2009). For example, we may have class labels or regression responses at our disposal. The goal of SDR is then to identify a subspace or other low-dimensional object that retains as much information as possible about the supervisory signal.

Having reduced dimensionality in this way, one may wish to subsequently build a classifier or regressor in the reduced representation. But there are other goals for the dimension reduction as well, including visualization, domain understanding, and domain transfer (i.e., predicting a different set of labels or responses).

In this chapter, we aim to combine these two lines of research and consider a supervised form of LDA. In particular, we wish to incorporate side information such as class labels into LDA, while retaining its favorable unsupervised dimensionality reduction abilities. The goal is to develop parameter estimation procedures that yield LDA topics that characterize the corpus and maximally exploit the predictive power of the side information.

As a parametric generative model, parameters in LDA are typically estimated by maximum likelihood estimation or Bayesian posterior inference. Such estimates are not necessarily optimal for yielding representations for prediction and regression. In this chapter, we use a discriminative learning criterion—conditional likelihood—to train a variant of the LDA model. Moreover, we augment the LDA parameterization by introducing class-label-dependent auxiliary parameters that can be tuned by the discriminative criterion. By retaining the original LDA parameters and introducing these auxiliary parameters, we are able to retain the interpretability advantages of the likelihood-based training procedure and provide additional freedom for tracking the side information.

The rest of the chapter is organized as follows. In Sec. 4.2, we introduce the discriminatively trained LDA (DiscLDA) model and contrast it to other related variants of LDA models. In Sec. 4.3, we describe our approach to parameter estimation for the DiscLDA model. In Sec. 4.4, we report empirical results on applying DiscLDA to model text documents. We discuss related work and some caveats about our model in Sec. 4.5. Finally, we present our conclusions in Sec. 4.6.

4.2 Model

We start by reviewing the LDA model (Blei et al., 2003) for topic modeling. We then describe our extension to LDA that incorporates class-dependent auxiliary parameters. These parameters are to be estimated based on supervised information provided in the training data set.

4.2.1 LDA

The LDA model is a generative process where each document in the text corpus is modeled as a set of draws from a mixture distribution over a set of hidden topics. A topic is modeled as a probability distribution over words. A *document* d is modeled as an exchangeable sequence of N_d words denoted by $\mathbf{w}_d = (w_{d1}, \dots, w_{dN_d})$. The generative process for this vector is illustrated in Fig. 4.1 and has three steps:

1. the document is first associated with a K -dimensional topic mixing vector $\boldsymbol{\theta}_d$ which is drawn from a Dirichlet distribution, $\boldsymbol{\theta}_d \sim \text{Dir}(\alpha)$;
2. before generating each word w_{dn} , we first choose its topic z_{dn} drawn from the multinomial variable, $z_{dn} \sim \text{Multi}(\boldsymbol{\theta}_d)$;
3. finally, the word w_{dn} is drawn from the corresponding topic $\boldsymbol{\phi}_{z_{dn}}$; it is a V -dimensional multinomial variable, $w_{dn} \sim \text{Multi}(\boldsymbol{\phi}_{z_{dn}})$, where V is the size of the vocabulary.

Given a set of documents, $\{\mathbf{w}_d\}_{d=1}^D$, the principal task is to estimate the parameters $\{\boldsymbol{\phi}_k\}_{k=1}^K$. This can be done by maximum likelihood, $\boldsymbol{\Phi}^* = \arg \max_{\boldsymbol{\Phi}} p(\{\mathbf{w}_d\}; \boldsymbol{\Phi})$, where $\boldsymbol{\Phi} \in \mathbb{R}^{V \times K}$ is a matrix parameter whose columns $\{\boldsymbol{\phi}_k\}_{k=1}^K$ are constrained to be members of a probability simplex. It is also possible to place a prior probability distribution on the word probability vectors $\{\boldsymbol{\phi}_k\}_{k=1}^K$ —e.g., a Dirichlet prior,

$\phi_k \sim \text{Dir}(\beta)$ —and treat the parameter Φ as well as the hyperparameters α and β via Bayesian methods. In both the maximum likelihood and Bayesian framework it is necessary to integrate over θ_d to obtain the marginal likelihood, and this is accomplished either using variational inference or Gibbs sampling (Blei et al., 2003; Griffiths and Steyvers, 2004).

4.2.2 DiscLDA

In the setting we wish to consider, each document is additionally associated with a categorical variable or class label $y_d \in \{1, 2, \dots, C\}$ (encoding, for example, whether a message was posted in the newsgroup `alt.atheism` vs. `talk.religion.misc`). To model this labeling information, we introduce a simple extension to the standard LDA model. Specifically, for each class label y , we introduce a linear transformation $\mathbf{T}^y : \mathfrak{R}^L \rightarrow \mathfrak{R}_+^K$, which transforms a L -dimensional Dirichlet variable θ_d to a mixture of K -dimensional Dirichlet distributions: $\theta_{\text{tr},d} = \mathbf{T}^y \theta_d \in \mathfrak{R}^K$. The rest of the generating process is then the same as in the standard LDA model, but using this transformed Dirichlet variable $\theta_{\text{tr},d}$ as the document prior for the topic indicator u_{dn} for word w_{dn} :

$$\begin{aligned} u_{dn} \mid \theta_d, y_d, \mathbf{T}^{y_d} &\sim \text{Multi}(\mathbf{T}^{y_d} \theta_d) \\ w_{dn} \mid u_{dn}, \Phi &\sim \text{Multi}(\phi_{u_{dn}}) \end{aligned} \tag{4.1}$$

It will become clear later in this section why we chose to change notation from z_{dn} to u_{dn} . Note that \mathbf{T}^y is constrained to have its columns sum to one to ensure the normalization of the transformed variable $\mathbf{T}^y \theta_d$ and can thus be interpreted as a stochastic matrix. Intuitively, every document in the text corpus is represented through θ_d as a point in the topic simplex $\{\theta \mid \sum_k \theta_k = 1\}$, and we hope that the linear transformations $\{\mathbf{T}^y\}$ will be able to *reposition* these points such that documents with the different class labels are represented by points far from each other — see Fig. 4.4.

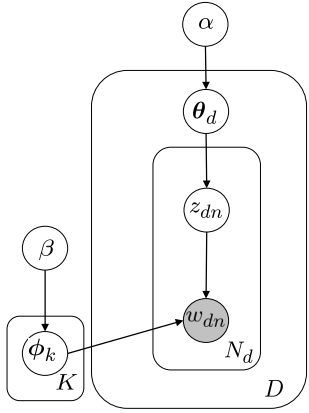
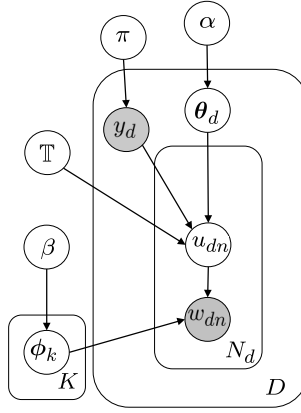
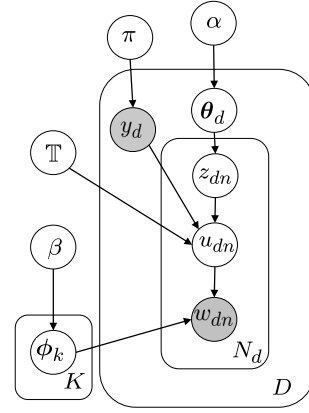


Figure 4.1: LDA model.


 Figure 4.2: DiscLDA model – this is for the interpretation where \mathbf{T}^y is applied to θ .

 Figure 4.3: DiscLDA with auxiliary variable z included. Marginalizing u or z yields the two different interpretations for the application of the linear transformation \mathbf{T}^y . This fixes the notation that we will use for the rest of the chapter.

Note that these points can *not* be placed arbitrarily, as all documents—whether they have the same class labels or they do not—share the parameter $\Phi \in \mathfrak{R}^{V \times K}$. The graphical model in Fig. 4.2 shows the new generative process. Compared to standard LDA, we have added the nodes for the variable y_d (and its prior distribution π), the set of transformation matrices $\{\mathbf{T}^y\}$ that we denote by \mathbb{T} , and the corresponding edges.

An alternative way to include the class information into LDA would be a model in which there are class-dependent topic parameters ϕ_k^y which determine the conditional distribution of the words:

$$w_{dn} \mid z_{dn}, y_d, \{\Phi^y\} \sim \text{Multi}(\phi_{z_{dn}}^{y_d}). \quad (4.2)$$

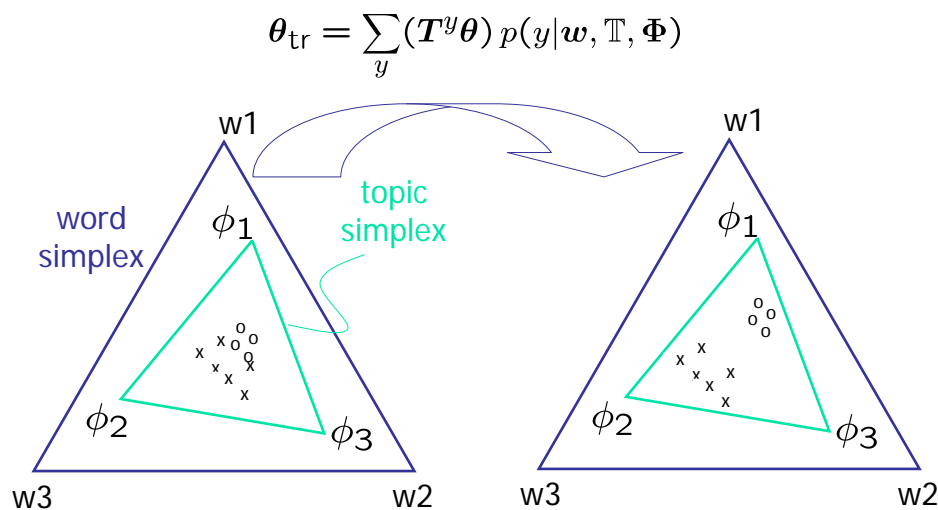


Figure 4.4: (Cartoon intuition) Each document can be represented on the topic mixture simplex $\{\theta \mid \sum_k \theta_k = 1\}$, yielding points $\Phi\theta$ on the word simplex. Here, the two classes are represented with circles and crosses. The hope is that the linear transformations $\{\mathbf{T}^y\}$ can reposition the points in such a way that the transformed representation θ_{tr} yields a better separation between classes.

The problem with this approach is that the posterior $p(y_d|\mathbf{w}_d, \{\Phi^y\})$ is a highly non-convex function of $\{\Phi^y\}$ which makes its optimization very challenging given the high dimensionality of the parameter space in typical applications (e.g. 50k words times 100 topics). Our approach circumvents this difficulty by learning a class-dependent low-dimensional *transformation* of some shared ϕ_k 's in a discriminative manner instead. Indeed, transforming the topic mixture vector θ is actually equivalent to transforming the Φ matrix. To see this, note that by marginalizing out the hidden

topic vector \mathbf{u} , we get the following distribution for the word w_{dn} given θ :

$$w_{dn} \mid \theta_d, y_d, \mathbb{T}, \Phi \sim \text{Multi}(\Phi \mathbf{T}^y \theta_d).$$

By the associativity of the matrix product, we see that we obtain an equivalent probabilistic model by applying the linear transformation to Φ instead, and, in effect, defining the class-dependent topic parameters as follows:

$$\phi_k^y = \sum_l \phi_l T_{lk}^y, \quad (4.3)$$

and then use them in a standard LDA model. We then obtain an analog to (4.2) for our DiscLDA model:

$$w_{dn} \mid z_{dn}, y_d, \mathbb{T}, \Phi \sim \text{Multi}(\phi_{z_{dn}}^{y_d}). \quad (4.4)$$

We are now in place to explain the difference between the variables \mathbf{z} and \mathbf{u} . We can give a generative interpretation to the transformation by augmenting the model with a hidden topic vector variable \mathbf{z} , as shown in Fig. 4.3, where

$$p(u_{dn} = k \mid z_{dn} = l, y_d, \mathbb{T}) = T_{kl}^{y_d}.$$

In this augmented model, \mathbf{T}^y can be interpreted as the probability transition matrix from z -topics to u -topics. The z -topic indicator represents the *prior* (untransformed) topic choice (which is marginally independent of y), whereas the u -topic indicator represents the transformed topic choice (which depends on y). In this notation, z has the same prior distribution as in the standard LDA, explaining our notational change mentioned before. An advantage of this new representation is that all variables now have conjugate priors, hence making Gibbs sampling straightforward to implement. Marginalization of z or u gives rise to non-conjugate priors because of the linearly

transformed Dirichlet variables. Moreover, to each marginalization corresponds one of the interpretations of the transformation: by marginalizing out z , we get the interpretation where \mathbf{T}^y is applied to $\boldsymbol{\theta}$, as in (4.1). On the other hand, marginalizing out u yields the interpretation where \mathbf{T}^y is applied to the class-independent parameters Φ as mentioned in equation (4.3), yielding the word distribution given in equation (4.4). Note that those two models yield the same joint distribution over words, when all latent variables are marginalized. Finally, by including a Dirichlet prior on the \mathbf{T}^y parameters, we could now take a full generative Bayesian approach and integrate it out or fix it to its MAP value using Gibbs sampling. We decided to focus on the discriminative treatment in this work, however.

Another motivation for our approach is that it gives the model the ability to distinguish topics which are shared across different classes versus topics which are class-specific. For example, this separation can be accomplished by using the following transformations (for binary classification):

$$\mathbf{T}^1 = \begin{pmatrix} \mathbf{I}_{B_c} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{B_s} \end{pmatrix}, \quad \mathbf{T}^2 = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{I}_{B_c} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{B_s} \end{pmatrix} \quad (4.5)$$

where \mathbf{I}_B stands for the identity matrix with B rows and columns. In this case, the last B_s topics are shared by both classes, whereas the two first groups of B_c topics are exclusive to one class or the other. We will explore this parametric structure later in our experiments.

4.3 Inference and learning

The general idea behind our learning procedure is the following. Given a corpus of documents and their labels, we estimate the parameters $\{\mathbf{T}^y\}$ discriminatively by

maximizing the conditional likelihood $\sum_d \log p(y_d | \mathbf{w}_d, \mathbb{T}, \Phi)$ while holding Φ fixed. To estimate the parameters Φ , we hold the transformation matrices fixed and maximize the posterior of the model, in much the same way as in standard LDA models. Intuitively, the two different training objectives have two effects on the model: the optimization of the posterior with respect to Φ captures the topic structure that is shared in documents throughout a corpus, while the optimization of the conditional likelihood with respect to $\{\mathbf{T}^y\}$ finds a transformation of the topics that discriminates between the different classes within the corpus.

However, the precise optimization algorithm depends on a few observations. First of all, none of the above steps can be done exactly as inference in our model is intractable (as in the LDA model). In addition, both the posterior over Φ and the conditional likelihood over \mathbf{y} are non-convex. This means that our results will depend on the precise way we decide to explore the space. Finally, following the convention used by [Griffiths and Steyvers \(2004\)](#) and [Blei et al. \(2003\)](#), we put a uniform Dirichlet prior over θ and ϕ_k (i.e., α and β are scalar constants). This means that the parameters in Φ are non-identifiable due to the invariance of the distribution to permutation of the indices $1, \dots, K$. Taking these facts into consideration, the general idea of our optimization algorithm is the following. We first get a good initial generative estimate of Φ by approximately maximizing its posterior for some initial plausible choice of $\{\mathbf{T}^y\}$. We then do *local* fine-tuning by doing gradient ascent with respect to $\{\mathbf{T}^y\}$ of the conditional likelihood on \mathbf{y} , for a fixed Φ . We could actually stop here (i.e., just optimize over $\{\mathbf{T}^y\}$ and not update Φ), but we actually get better results by iteratively updating Φ given this $\{\mathbf{T}^y\}$ as well. However, just maximizing the posterior on the high-dimensional Φ could move us to a far away local minimum obtained by permuting indices. We thus restrict the update on Φ to be *local* as well by conditioning on the previous value of Φ . The algorithm is outlined in [Table 4.1](#), and described in the next section. But the general idea of our

1. Initialize \mathbb{T}_0 to a set of smoothed block diagonal matrices.
2. (a) Sample N RB Gibbs steps from $p(\mathbf{u}, \mathbf{z}|\mathbf{y}, \mathbf{w}, \mathbb{T}_0)$ [use (4.25)].
 (b) Use the last sample $\mathbf{u}^{(N)}$ to set Φ to its posterior mean [see (4.27)]:

$$\Phi_0 = \mathbb{E} [\Phi | \mathbf{u}^{(N)}, \mathbf{w}]. \quad (4.6)$$

3. Repeat until no improvement on the validation set:
 - (a) Repeat N_g steps:

$$\mathbf{T}_{t+1}^{y'} = \mathbf{T}_t^{y'} + \eta \sum_d \frac{\partial}{\partial \mathbf{T}^{y'}} \log p(y_d | \mathbf{w}_d, \mathbb{T}_t, \Phi_t) \quad \text{for } y' = 1, \dots, C. \quad (4.7)$$

[using equation (4.17) for $\frac{\partial}{\partial \mathbf{T}^{y'}} \log p(y_d | \mathbf{w}_d, \mathbb{T}_t, \Phi_t)$].

- (b) Sample N RB Gibbs steps from $p(\mathbf{z}|\mathbf{y}, \mathbf{w}, \mathbb{T}_t, \Phi_t)$ [use (4.22)].
 - (c) Sample 1 Gibbs step from $p(\mathbf{u}|\mathbf{z}^{(N)}, \mathbf{y}, \mathbf{w}, \mathbb{T}_t, \Phi_t)$ [use (4.24)].
 - (d) Set $\Phi_{t+1} = \mathbb{E} [\Phi | \mathbf{u}^{(1)}, \mathbf{w}]$ [see (4.27)].
4. Return (\mathbb{T}_t, Φ_t) with the minimum validation prediction error.

Table 4.1: DiscLDA learning algorithm. The constant step-size η is chosen by minimizing the prediction error on the validation set. The notes in brackets refer to the detailed equations provided in Appendix 4.A.

approach is to use the generative criterion to move us to a specific part of the space, and then discriminatively fine tune it, while keeping the Φ parameter consistent with the change in $\{\mathbf{T}^y\}$ through local updates.

4.3.1 Approximate inference for learning

Two main approaches for approximate inference have been used for probabilistic topic models: variational inference (Blei et al., 2003; Teh et al., 2007) or Gibbs sampling (Griffiths and Steyvers, 2004). In this work, we first decided to experiment

with the sampling approach, even though it would be interesting to compare it with a variational approach in subsequent work.

We use a Rao-Blackwellized version of Gibbs sampling similar to the one presented in Griffiths and Steyvers (2004) to obtain samples of (\mathbf{z}, \mathbf{u}) with (Φ, θ) marginalized out, for some initial $\mathbb{T} = \mathbb{T}_0$. As in Griffiths and Steyvers (2004), we get a generative point estimate of Φ by setting it to its posterior mean given the last sample $\mathbf{u}^{(N)}$, where N is large enough to ensure proper mixing of the chain (step 2 of the algorithm)¹. We provide the detailed equations in Appendix 4.A.2. After this initial estimate, all subsequent updates are made *local* by conditioning on Φ (see step 3 of the algorithm). Apart locality, another advantage of conditioning on Φ is that it decouples the sampling for each document in independent updates, and makes it possible to use Gibbs sampling to sample from $p(\mathbf{z}|\mathbf{y}, \mathbf{w}, \mathbb{T}_t, \Phi_t)$ (which has identical updates to a collapsed Gibbs sampler for LDA with class-dependent parameters). This is used in step 3b and 3c to obtain $\mathbf{u}^{(N)}$ in a faster fashion than jointly sampling (\mathbf{u}, \mathbf{z}) as in step 2a – which is quadratically slower than just sampling \mathbf{z} ², as is explained in Appendix (4.A.2).

The gradient of the conditional likelihood objective with respect to \mathbf{T}^y is estimated using Monte Carlo EM, with samples from the Gibbs sampler conditioned on Φ . More specifically, as we will describe more in details in Appendix 4.A, we use the matching

¹The motivation for the posterior mean as a point estimate of Φ is that it represents the parameter of the Bayesian predictive distribution for one word in a new test document given the training set and the training topic assignment $\mathbf{u}^{(N)}$. Ideally, the point estimate for Φ would be obtained by averaging several samples from the posterior over \mathbf{u} . Unfortunately, as columns of Φ are non-identifiable under permutation, different samples could correspond to different permutations and so cannot be combined. The standard approach is thus to use only one Gibbs sample, which is a (poor) stochastic approximation of the MAP estimate of $p(\mathbf{u}, \mathbf{z}|\mathbf{y}, \mathbf{w}, \mathbb{T})$.

²Note that Gibbs sampling of $p(\mathbf{u}|\mathbf{y}, \mathbf{w}, \mathbb{T}_t, \Phi_t)$ (\mathbf{z} marginalized out) is not possible due to non-conjugacy – θ can no longer be analytically integrated out due to the mixing of Dirichlet variables as in equation (4.1). This computational restriction explains why we are sampling from the joint posterior on (\mathbf{u}, \mathbf{z}) rather than just the marginal posterior on \mathbf{u} – which would seem more natural to estimate Φ .

property of gradients in EM to write the gradient as:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{T}^{y'}} \log p(y_d | \mathbf{w}_d, \mathbb{T}, \Phi) &= \mathbb{E}_{q_t^{y_d}(\mathbf{z}_d)} \left[\frac{\partial}{\partial \mathbf{T}^{y'}} \log p(\mathbf{w}_d, \mathbf{z}_d | y_d, \mathbb{T}, \Phi) \right] \\ &\quad - \mathbb{E}_{r_t(\mathbf{z}_d)} \left[\frac{\partial}{\partial \mathbf{T}^{y'}} \log p(\mathbf{w}_d, \mathbf{z}_d | \mathbb{T}, \Phi) \right], \end{aligned} \quad (4.8)$$

where $q_t^{y_d}(\mathbf{z}_d) = p(\mathbf{z}_d | y_d, \mathbf{w}_d, \mathbb{T}_t, \Phi)$, $r_t(\mathbf{z}_d) = p(\mathbf{z}_d | \mathbf{w}_d, \mathbb{T}_t, \Phi)$ and the derivatives are evaluated at $\mathbb{T} = \mathbb{T}_t$. The terms inside the expectation can be computed analytically, and we can approximate the integrals using our Gibbs samples (see Appendix 4.A.1.2). However, it turns out in our experiments that sampling from $r_t(\mathbf{z}_d) = p(\mathbf{z}_d | \mathbf{w}_d, \mathbb{T}_t, \Phi)$ is hard as the chain doesn't mix well: \mathbf{z}_d often yields a very peaked posterior $p(y_d | \mathbf{z}_d, \mathbf{w}_d, \mathbb{T}, \Phi)$ (for example, with maximum at y_d^*), and thus in effect we get stuck with samples from $p(\mathbf{z}_d | y_d^*, \mathbf{w}_d, \mathbb{T}_t, \Phi)$ from whatever value of y_d^* at which we converged from our initial random initialization of \mathbf{z}_d . For this reason, we analytically integrate out y_d in the second expectation of equation (4.8) by using an estimate of the posterior $p(y_d | \mathbf{w}_d, \mathbb{T}, \Phi)$.

4.3.2 Approximating the posterior over labels

To do prediction in our model or to approximate the gradient as we mentioned above, we need to approximate the posterior $p(y_d | \mathbf{w}_d, \mathbb{T}, \Phi)$. We do this by using the bridge sampling method of [Meng and Wong \(1996\)](#), a standard method in Bayesian statistics to estimate ratio of normalizing constants. We provide background material for bridge sampling in Appendix 4.B. We mention alternatives and provide the details for our setup in Appendix 4.B.1³. As an overview, we can express the likelihood ratios of

³Note that we had used the harmonic mean estimator instead (as in [Griffiths and Steyvers \(2004\)](#)) in the preliminary version of this work which appeared in [Lacoste-Julien et al. \(2009\)](#). For a fixed \mathbf{T}^y , the harmonic mean estimator appeared to give stable predictions in our initial experiments. We only realized later that it gives significantly noisy estimates when analyzed carefully in the context of discriminatively learned \mathbf{T}^y .

two classes y_1, y_0 as:

$$\frac{p(y_1|\mathbf{w}_d, \mathbb{T}, \Phi)}{p(y_0|\mathbf{w}_d, \mathbb{T}, \Phi)} = \frac{p(y_1)}{p(y_0)} \frac{Z_1}{Z_0}$$

where Z_c is the normalization constant for $q_c(\mathbf{z}_d) \doteq p(\mathbf{w}_d|\mathbf{z}_d, y_c, \mathbb{T}, \Phi)p(\mathbf{z}_d)$ which is proportional to the posterior on \mathbf{z}_d given y_c . We use bridge sampling with a geometric bridge between these two posteriors to estimate the likelihood ratio of each class to a fixed reference class (the last one in our experiments) - see equation (4.33). We can obtain the final posterior on y by renormalizing these ratios. To compute the bridge sampling estimate, we simply need samples of $p(\mathbf{z}_d|y, \mathbf{w}_d, \mathbb{T}, \Phi)$ for each possible class y , which can be obtained via Rao-Blackwellized Gibbs sampling (see (4.22)).

4.3.3 Dimensionality reduction

We can obtain a supervised dimensionality reduction method by using the average transformed topic vector as the reduced representation of a test document. We estimate it using

$$\mathbb{E}[\mathbf{T}^y \boldsymbol{\theta}_d | \mathbf{w}_d, \mathbb{T}, \Phi] = \sum_y p(y | \mathbf{w}_d, \mathbb{T}, \Phi) \mathbf{T}^y \mathbb{E}[\boldsymbol{\theta}_d | y, \mathbf{w}_d, \mathbb{T}, \Phi]. \quad (4.9)$$

The first term on the right-hand side of this equation can be approximated using the bridge sampling estimator and the last term can be approximated from MCMC samples of \mathbf{z}_d (in the outer expectation on the right of the following equation):

$$\mathbb{E}[\boldsymbol{\theta}_d | y, \mathbf{w}_d, \mathbb{T}, \Phi] = \mathbb{E}[\mathbb{E}[\boldsymbol{\theta}_d | \mathbf{z}_d] \mid y, \mathbf{w}_d, \mathbb{T}, \Phi]. \quad (4.10)$$

See Appendix 4.A.3 for the details. The inner expectation is a simple posterior mean for a Dirichlet distributed variable. This new representation can be used as a feature vector for another classifier or for visualization purposes.

4.4 Experimental results

We evaluated the DiscLDA model empirically on text modeling and classification tasks. Our experiments aimed to demonstrate the benefits of discriminative training of LDA for discovering a compact latent representation that contains both predictive and shared components across different types of data. We evaluated the performance of our model by contrasting it to standard LDA models that were not trained discriminatively⁴.

4.4.1 Dataset description

For our experiments, we use the *20 Newsgroups* dataset, which contains postings to Usenet newsgroups. The postings are organized by content into 20 related categories and are therefore well suited for topic modeling. We use the train-test split from the standard Matlab file available from <http://people.csail.mit.edu/jrennie/20Newsgroups/> and removed a standard list of stop words⁵. We consider the classification task for the whole 20 Newsgroups as well as five smaller binary classification tasks of challenging pairs of classes. The statistics of the datasets we consider is given in Table 4.2. The vocabulary size is 62k for all our experiments. We also use a random split of 20% of the training set as our validation set. Note that the standard train-test split is more challenging than random splits as they span two different time periods of postings.

⁴Our implementation is available from <http://www.cs.berkeley.edu/~slacoste/research/discLDA/>.

⁵We used the list of 490 stop words available from the class `mallet-0.4/src/edu/umass/cs/mallet/base/pipe/TokenSequenceRemoveStopwords.java` from the MALLET package available at <http://mallet.cs.umass.edu/>.

dataset	split	# of documents	# of tokens
20 Newsgroups	train	11,269	1,3M
	test	7,500	860k
Binary classification tasks:			
1 vs. 20: alt.atheism vs. talk.religion.misc	train	856	105k
	test	569	80k
16 vs. 20: soc.religion.christian vs. talk.religion.misc	train	975	115k
	test	649	100k
4 vs. 5: comp.sys.ibm.pc.hardware vs. comp.sys.mac.hardware	train	1,162	110k
	test	775	55k
4 vs. 3: comp.sys.ibm.pc.hardware vs. comp.os.ms-windows.misc	train	1,159	115k
	test	783	70k
10 vs. 11: rec.sport.baseball vs. rec.sport.hockey	train	1,191	115k
	test	796	80

Table 4.2: Properties of the 20 Newsgroups dataset and several binary pairs we chose.

4.4.2 Experiments with fixed \mathbf{T}^y

4.4.2.1 Text modeling

In this section, we first investigate how DiscLDA can exploit the labeling information—the category—in discovering meaningful hidden structures that differ from those found using unsupervised techniques, for the case where \mathbf{T}^y is fixed and chosen by us. This enables us to explore the benefits of identifying shared topics and class-specific topics, while avoiding the computational burden of discriminatively learning \mathbf{T}^y , which we consider in Sec. 4.4.3.

We fit the dataset to both a standard 110-topic LDA model and a DiscLDA model with restricted forms of the transformation matrices $\{\mathbf{T}^y\}_{y=1}^{20}$. Specifically, the transformation matrix \mathbf{T}^y for class label y is *fixed* and given by the following blocked

matrix:

$$\mathbf{T}^y = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \vdots & \vdots \\ \mathbf{I}_{B_c} & \mathbf{0} \\ \vdots & \vdots \\ \mathbf{0} & \mathbf{I}_{B_s} \end{pmatrix}. \quad (4.11)$$

This matrix has $(C + 1)$ rows and two columns of block matrices. All but two block matrices are zero matrices. At the first column and the row y , the block matrix is an identity matrix with dimensionality of $B_c \times B_c$. The last element of \mathbf{T}^y is another identity matrix with dimensionality B_s . When applying the transformation to a topic vector $\boldsymbol{\theta} \in \Re^{B_c+B_s}$, we obtain a transformed topic vector $\boldsymbol{\theta}_{\text{tr}} = \mathbf{T}^y \boldsymbol{\theta}$ whose nonzero elements partition the components $\boldsymbol{\theta}_{\text{tr}}$ into $(C+1)$ disjoint sets: one set of B_c elements for each class label that does not overlap with the others (the *class-dependent* topic parameters), and a set of B_s components that is *shared* by *all* class labels (the *shared* topic parameters). Intuitively, the shared components should use all class labels to model common latent structures, while nonoverlapping components should model specific characteristics of data from each class.

In a first experiment, we examined whether the DiscLDA model can exploit the structure for \mathbf{T}^y given in (4.11). In this experiment, we first obtained an estimate of the Φ matrix by setting it to its posterior mean conditioned on a topic assignment vector $\mathbf{u}^{(N)}$ obtained after $N = 200$ steps of Gibbs sampling⁶ – see equation (4.6) in step 2 of the algorithm shown in Table 4.1. This is in a similar fashion as in Griffiths and Steyvers (2004). Note that we cannot use multiple Gibbs samples to estimate Φ , as columns are non-identifiable under permutation – different samples could cor-

⁶We studied the autocorrelation function (see e.g. Liu, 2001) for several quantities of interest based on the samples of \mathbf{u} (e.g. the log-likelihood $\log p(\mathbf{w}_d | \mathbf{u}^{(i)})$, or the posterior mean for $\boldsymbol{\theta}_d$ for different d 's), and observed that the autocorrelation would decrease close to zero for a lag varying between 10 to 50. $N = 200$ is thus a conservative number. We used the same burn-in for the LDA model and for the bridge sampling estimate.

respond to different permutations. We then estimated a new representation for test documents by taking the conditional expectation of $\mathbf{T}^y \boldsymbol{\theta}$ with y marginalized out as explained in the dimensionality reduction Sec. 4.3.3. Note that the class label of the test document is obviously not needed in the computation of its new representation. We used bridge sampling with $M = 2.5k$ samples after 200 steps of burn-in each time we need an estimate of $p(y|\mathbf{w}_d, \mathbb{T}, \Phi)$ – see equation (4.33). Finally, we computed a 2D-embedding of this K -dimensional representation of documents ($K = CB_c + B_s$) to investigate its properties. The first embedding we tried was to run standard multidimensional scaling (MDS), using the symmetrical KL divergence between pairs of $\boldsymbol{\theta}_{\text{tr}}$ topic vectors as a dissimilarity metric, but the results were hard to visualize. A more interpretable embedding was obtained using the t-SNE stochastic neighborhood embedding method presented by [van der Maaten and Hinton \(2008\)](#)⁷. Fig. 4.5 shows a scatter plot of the 2D-embedding of the topic representation of the *20 Newsgroups* test documents, where the colors and the shape of the dots, each corresponding to a document, encode different groupings and different classes within a group, respectively. Clearly, the documents are well separated in this space. In contrast, the embedding computed from standard LDA, shown in Fig. 4.6, does not show a clear separation. In this experiment, we have set $B_c = 5$ and $B_s = 10$ for DiscLDA, yielding $K = 110$ possible topics; hence we set $K = 110$ for the standard LDA model for proper comparison. For all our experiments, we used $\beta = 0.01$ and $\alpha = 10/K$.

It is also instructive to examine in detail the topic structures of the fitted DiscLDA model. Given the specific setup of our transformation matrix \mathbf{T}^y , each component of the topic vector \mathbf{u} is either associated with a class label or shared across all class labels. For each component, we can compute the most popular words associated from the word-topic distribution Φ . In Table 4.3, we list these words and group them

⁷We used the “fast t-SNE” C implementation available from http://ticc.uvt.nl/~lvdrmaaten/Laurens_van_der_Maaten/t-SNE.html with its default options.

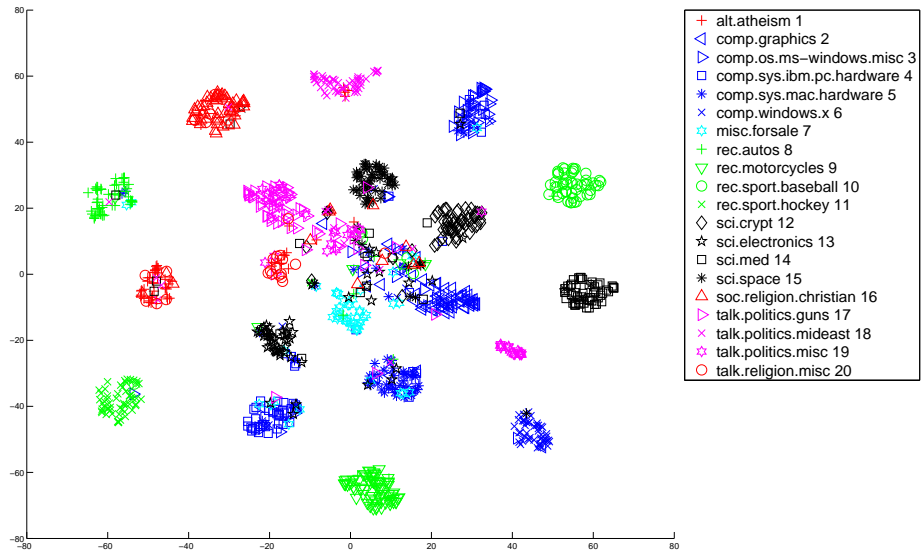


Figure 4.5: t-SNE 2D embedding of the $\mathbb{E} [T^y \theta_d | w_d, T, \Phi]$ representation of Newsgroups documents, after fitting to the DiscLDA model (T^y was fixed).

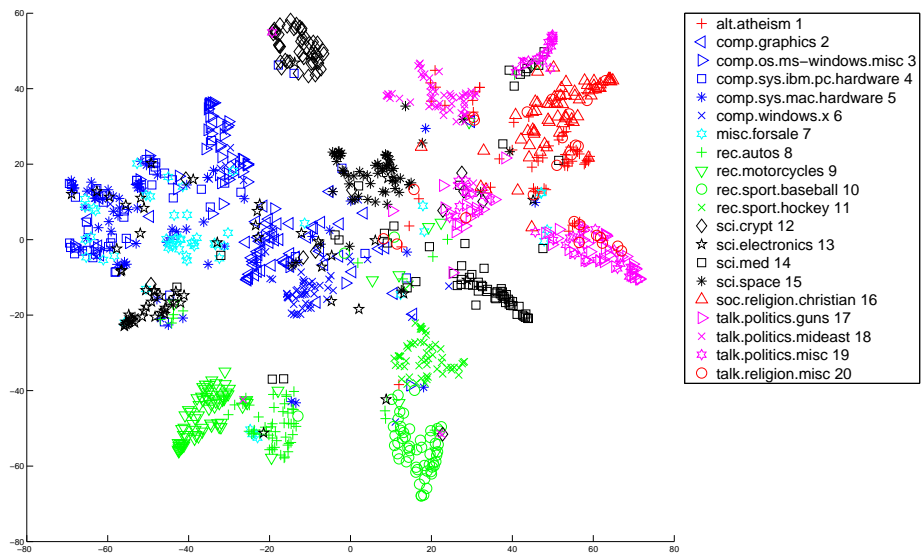


Figure 4.6: t-SNE 2D embedding of the $\mathbb{E} [\theta_d | w_d, \Phi]$ representation of Newsgroups documents, after fitting to the standard unsupervised LDA model.

under each class labels and a special bucket “shared”. The sorted list of words for each group G was obtained by computing the following quantity for each word v : $s_v^G \doteq \frac{V}{|G|} \sum_{k \in G} \Phi_{vk}$, which represents roughly how more likely than uniform the word is in this group G . We see that the words are indicative of their associated class labels. Additionally, the words in the “shared” category are “neutral,” neither positively nor negatively suggesting proper class labels where they are likely to appear. In fact, these words confirm the intuition of the DiscLDA model: they reflect common English usage underlying this corpus.

4.4.2.2 Document classification

It is also of interest to consider the classification problem more directly and ask whether the features delivered by DiscLDA are more useful for classification than those delivered by LDA. Of course, we can also use DiscLDA as a classification method per se, by marginalizing over the latent variables and computing the probability of the label y given the words in a test document. Our focus in this section, however, is its featural representation. We thus use a different classification method (the SVM) to compare the features obtained by DiscLDA to those obtained from LDA.

Continuing our first experiment, we returned to the fixed \mathbf{T}^y setting studied in Sec. 4.4.2.1 and considered the features obtained by DiscLDA for the *20 News-groups* problem. Specifically, we constructed multiclass linear SVM classifiers using the expected topic proportion vectors from unsupervised LDA and DiscLDA models as features as described in Sec. 4.3.3. We deliberately chose not to use kernels for the SVM classifiers, as we want to focus on whether a *simple* classification can be done from the reduced representation. We used the SVM^{multiclass} implementation of SVM^{struct} available from http://svmlight.joachims.org/svm_struct.html. The regularization parameter C for the SVM is chosen by grid search on the validation

Chapter 4. DiscLDA: Discriminative Dimensionality Reduction for Classification

1 alt. atheism	2 comp. graphics	3 comp.os.ms- windows. misc	4 comp.sys. ibm.pc. hardware	5 comp.sys. mac. hardware	6 comp. windows.x	7 misc. forsale
god: 9 writes: 6 islam: 4 keith: 4 religion: 4 atheism: 4 article: 3 morality: 3 moral: 3 bible: 3	graphics: 9 image: 8 file: 4 jpeg: 3 op: 3 files: 3 images: 3 color: 2 compass: 2 points: 2	windows: 18 file: 6 dos: 5 files: 5 mouse: 4 win: 4 driver: 3 ms: 3 program: 3 card: 3	drive: 13 scsi: 12 mb: 7 card: 7 disk: 6 controller: 6 ide: 6 drives: 5 system: 5 bus: 5	apple: 10 mac: 10 drive: 6 mb: 4 scsi: 4 bit: 3 mhz: 3 simms: 3 problem: 3 cd: 3	window: 7 file: 5 entry: 4 program: 4 output: 4 motif: 3 code: 3 widget: 3 server: 3 lib: 3	sale: 6 shipping: 5 dos: 4 offer: 4 condition: 4 cover: 3 excellent: 3 st: 3 cd: 3 price: 2
8 rec.autos	9 rec. motorcycles	10 rec.sport. baseball	11 rec.sport. hockey	12 sci.crypt	13 sci. electronics	14 sci.med
car: 16 cars: 7 engine: 4 writes: 3 article: 2 dealer: 2 speed: 2 drive: 2 driving: 2 oil: 2	bike: 11 dod: 7 ride: 4 bikes: 3 helmet: 3 motorcycle: 3 riding: 3 dog: 3 bmw: 2 article: 2	year: 6 game: 5 baseball: 5 team: 5 games: 4 season: 3 cubs: 3 players: 3 runs: 3 league: 2	team: 7 hockey: 5 game: 5 play: 5 players: 4 nhl: 3 win: 3 season: 3 period: 3 teams: 2	db: 11 key: 11 encryption: 5 chip: 5 security: 4 privacy: 4 clipper: 4 keys: 4 anonymous: 3 government: 3	ground: 4 wire: 4 power: 3 circuit: 3 wiring: 3 current: 2 voltage: 2 water: 2 amp: 2 output: 2	msg: 4 medical: 3 food: 3 pitt: 3 disease: 3 health: 3 article: 2 patients: 2 gordon: 2 science: 2
15 sci.space	16 soc.religion. christian	17 talk.politics. guns	18 talk.politics. mideast	19 talk.politics. misc	20 talk.religion. misc	Shared topics
space: 13 nasa: 6 launch: 5 moon: 4 earth: 3 orbit: 3 lunar: 3 henry: 3 shuttle: 2 satellite: 2	god: 18 church: 6 jesus: 6 christians: 5 bible: 5 christian: 5 christ: 5 faith: 4 truth: 4 hell: 4	gun: 10 file: 5 guns: 5 weapons: 4 firearms: 4 militia: 3 government: 3 fire: 3 weapon: 3 control: 3	armenian: 6 israel: 5 turkish: 6 armenians: 4 jews: 4 israeli: 4 people: 3 turkey: 2 armenia: 2 turks: 2	government: 5 president: 4 mr: 4 article: 4 stephanopoulos: 3 writes: 3 war: 3 health: 3 cramer: 3 tax: 2	god: 7 jesus: 7 christian: 4 bible: 4 objective: 3 sandvik: 3 morality: 3 christ: 3 christians: 2 ra: 2	writes: 9 article: 7 don: 6 people: 5 time: 4 apr: 4 good: 3 make: 3 ve: 3 ca: 2

Table 4.3: Ten most popular words from each group of class-dependent topics or a bucket of “shared” topics learned in the *20 Newsgroups* experiment with fixed T^y matrix. The number after each word is s_v^G and represents roughly how more likely than uniform the word v is in the group G (rounded in units of hundreds here to save space).

LDA+SVM	DiscLDA+SVM	DiscLDA alone
27.0%	19.2%	19.2%

Table 4.4: Classification error rates on 20 Newsgroups dataset where DiscLDA uses a fixed \mathbf{T}^y matrix chosen by us.

set⁸. The results are summarized in Table 4.4. Going from LDA to DiscLDA as the dimensionality reduction method reduced the error rate from 27.0% to 19.2%, indicating that the DiscLDA features have retained information useful for classification. For comparison, we also computed the MAP estimate of the class label $y^* = \arg \max p(y|\mathbf{w}, \mathbb{T}, \Phi)$ from DiscLDA and used this estimate directly as a classifier. The DiscLDA classifier actually obtained a similar error rate as running SVM on the DiscLDA features, though this was not always the case for the other experiments we ran in the next section⁹.

4.4.3 Experiments with discriminatively trained \mathbf{T}^y

In this section, we consider the fully adaptive setting in which the transformation matrix \mathbf{T}^y is learned in a discriminative fashion as described in the full algorithm of Table 4.1. For simplicity of implementation (affected by scaling issues), we only experimented on binary classification tasks. We initialized the matrix \mathbf{T}^y to a smoothed block diagonal matrix¹⁰ having the same pattern as in equation (4.5), but with $C = 2$ classes. We used $B_s = 20$ shared topics and $B_c = 20$ class-dependent topics per class. We used $N = 200$ in steps 2 and 3 of the algorithm; and $N_g = 10$ for the number of gradient steps done before Φ is updated in the loop of step 3. The constant step size η was chosen by grid search (minimum validation error). The discriminative learning

⁸We search over a geometric grid from 10^{-3} to 10^9 .

⁹As a reference point, smoothed naive Bayes gets 19.9% on this split.

¹⁰We worked in the log-domain parameterization \mathbf{t}^y to ensure that \mathbf{T}^y stays normalized. The matrix \mathbf{T}_y shown in (4.5) lies at the boundary of this domain; we thus use the smoother initialization $\mathbf{t}_0^y = 3\mathbf{T}_y - 2$.

loop in step 3 was meant to be repeated until there was no improvement in accuracy on the validation set – though in practice, the error curve didn’t have a clear trend so we simply chose the point with minimal validation error. We used the same bridge sampling parameters as in the previous section.

In this experiment, we considered the five binary classification problems listed in Table 4.2. The choice of pair was motivated by their sharing of some thematic content, such as for example distinguishing postings of the newsgroup `alt.atheism` 1 from postings of the newsgroup `talk.religion.misc` 20, contributing to their difficulty (with the exception of `rec.sport.baseball` 10 vs. `rec.sport.hockey` 11 which is surprisingly easy).

We ran the same comparison of classification experiments on these datasets as in the previous section, with the exception that DiscLDA here refers to the fully discriminatively trained DiscLDA. The results are presented in Fig. 4.7. On almost all tasks, the features of DiscLDA yielded lower classification error vs. LDA when used as input to a SVM classifier. The exception is on tasks 10 vs. 11, for which the LDA + SVM classifier already obtained good accuracy.

As in the previous section, we can also explore the structure in the text uncovered by DiscLDA. It turns out that the discriminatively learned \mathbf{T}^y also possess a block-diagonal structure (though a different one than from the starting point – the gradient search did produce non-trivial changes to it). We can thus also identify “strongly” class-specific topics and “strongly” shared topics, where the strength of this identity is characterized by the sparsity of the sharing between the same rows of \mathbf{T}^y for the two different classes. As a heuristic way to identify these types, we consider $s_k^y \doteq \sum_l T_{kl}^y$ as the strength of the presence of topic k for class y (since both classes have the same uniform Dirichlet prior). We then choose to label the topics k for which $s_k^y \geq 0.9$ for both classes as the “strongly” shared topics; and the topics k for which $s_k^y \geq 0.9$ for one class but $s_k^y \leq 0.4$ for the other as the “strongly” class-specific topic for

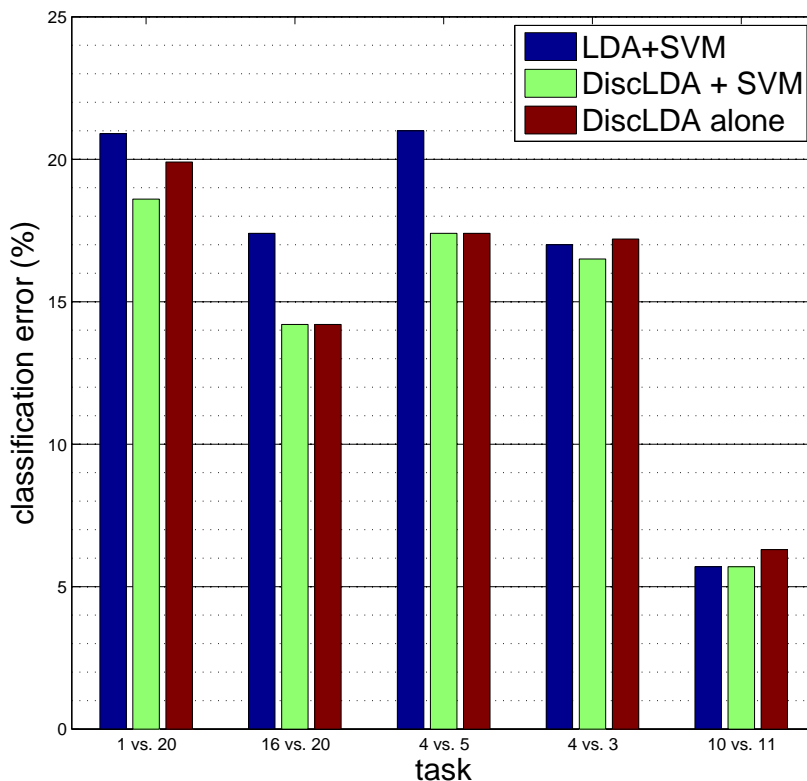


Figure 4.7: Classification error for several binary classification tasks. Note that DiscLDA here refers to the discriminatively trained version.

the former class. We present the ten most probable words for each of these groups learned by DiscLDA for the different tasks in Table 4.5, in analogy to Table 4.3. It is interesting to contrast the results for different tasks (even though the results are obviously noisy). For example, words like “point” and “evidence” appear to be class-specific for the `talk.religion.misc` 20 newsgroup when trying to distinguish these posts from the more purely religious newsgroup `soc.religion.christian` 16. On the other hand, these words (as well as “evidence” e.g.) appear to be in the shared group when the class 20 is compared with `alt.atheism` 1 (and *not* in the top words for the shared group of 16 vs. 20) – the class-specific group for 20 for the 1 vs.

20 task has more christianity words such as “christians, evil, bible”, discriminative words versus the `alt.atheism` newsgroup. These christianity words rather appear in the shared bucket for the 16 vs. 20 task, unsurprisingly. These interpretations have obviously to be taken with a grain of salt, as is typical in such exploratory analysis.

A more refined interpretation can be obtained by looking at the individual topics (rather than the group as a whole) – which we present in Table 4.6 for the 4 vs. 5 task. In this case, we only show the three dominant topics per group. The strength of a topic k for class c is heuristically computed as follows: take the average value of $\mathbb{E}[\sum_l T_{kl}^y \theta_{dl} | y_d, \mathbf{w}_d, \mathbb{T}, \Phi]$ over the training documents d 's with class $y_d = c$ – this represents the expected proportion of this topic k due only to the class c and is used as an indicator of class-dependent topic strength. To evaluate the strength of a topic in a shared group, we use the average value of this quantity for the two classes. As an example of observation we can make from Table 4.6, we see that a topic about monitors only appear as a dominant class-specific topic for `comp.sys.ibm.pc.hardware` 4 in the task of distinguishing it from OS type posts in `comp.os.ms-windows.misc` 3; it doesn't appear as a dominant class-specific topic for 4 in the 4 vs. 5 task, in which both classes contain hardware related posts.

Finally, we present for reference the t-SNE 2D-embeddings of the DiscLDA topic representation in comparison with LDA for these binary tasks in Fig. 4.8 and Fig. 4.9. The improvement over LDA is also clear here.

Chapter 4. DiscLDA: Discriminative Dimensionality Reduction for Classification

1 vs. 20: alt.atheism vs. talk.religion.misc			16 vs. 20: soc.religion.christian vs. talk.religion.misc		
class 1 (21)	class 20 (18)	Shared (7)	class 16 (8)	20 (18)	Shared (9)
don: 609	jesus: 458	people: 1168	god: 770	writes: 470	good: 803
writes: 548	god: 418	writes: 1074	church: 538	god: 466	christian: 717
god: 461	christian: 229	good: 1051	time: 463	christ: 371	people: 716
article: 219	man: 186	article: 861	hell: 346	article: 360	god: 605
people: 211	evil: 178	don: 712	people: 331	don: 308	love: 583
apr: 190	christ: 150	life: 657	work: 243	people: 281	christians: 577
system: 189	christians: 140	god: 496	resurrection: 238	objective: 165	jesus: 540
keith: 171	bible: 139	fact: 493	body: 231	sandvik: 165	man: 526
religion: 163	people: 137	point: 466	things: 202	point: 155	religion: 484
moral: 162	koresh: 136	evidence: 435	marriage: 197	evidence: 147	writes: 461

4 vs. 5: comp.sys.ibm.pc.hardware vs. comp.sys.mac.hardware			4 vs. 3: comp.sys.ibm.pc.hardware vs. comp.os.ms-windows.misc		
class 4 (18)	class 5 (17)	Shared (11)	class 4 (7)	class 3 (4)	Shared (12)
dos: 302	apple: 525	drive: 970	bit: 837	windows: 2530	scsi: 669
system: 294	problem: 336	mb: 682	mb: 800	article: 871	good: 491
don: 242	mac: 328	bit: 450	writes: 724	win: 604	don: 460
bus: 230	writes: 312	disk: 449	article: 609	find: 591	drive: 430
problem: 215	don: 268	drives: 378	bus: 594	nt: 581	card: 384
time: 215	ve: 249	writes: 366	data: 362	change: 422	system: 366
cards: 206	se: 214	system: 304	isa: 353	os: 362	memory: 357
apr: 200	monitor: 214	ram: 302	monitor: 334	ms: 320	mail: 344
drive: 198	computer: 210	software: 286	chip: 331	editor: 281	program: 298
motherboard: 194	article: 197	article: 284	fast: 286	text: 280	hard: 298

10 vs. 11: rec.sport.baseball vs. rec.sport.hockey		
class 10 (20)	class 11 (31)	Shared (5)
year: 373	team: 377	article: 1586
game: 291	play: 358	writes: 1216
baseball: 226	game: 352	games: 917
writes: 208	hockey: 258	don: 848
time: 194	time: 221	team: 848
runs: 182	win: 198	good: 575
hit: 178	players: 176	ca: 571
apr: 175	writes: 175	season: 544
team: 158	ca: 167	mike: 455
players: 157	year: 164	didn: 408

Table 4.5: Ten most popular words from each group of “strongly” class-dependent topics for different binary classification tasks on the *20 Newsgroups* with the T^y matrix learned discriminatively. A “strongly” shared topic group is also identified for each task. The number after each word represents roughly how more likely than uniform the word is in the group.

Chapter 4. DiscLDA: Discriminative Dimensionality Reduction for Classification

4 vs. 5 task					
topics for comp.sys.ibm.pc.hardware 4			topics for comp.sys.mac.hardware 5		
5.0	4.0	3.8	4.3	4.0	3.8
controller: 3455	irq: 2418	gateway: 1502	apple: 6681	sound: 1807	vram: 1744
bus: 3056	port: 2159	windows: 1349	fpu: 1757	system: 1762	chip: 1261
ide: 3018	mouse: 1497	drivers: 866	price: 1331	duo: 1762	apple: 1208
isa: 2221	pc: 1266	ati: 815	centris: 1145	problem: 1514	heard: 1127
pc: 2050	set: 1007	pro: 738	lc: 825	apple: 994	writes: 939
transfer: 1196	sx: 979	writes: 713	ve: 799	powerbook: 791	lciii: 913
speed: 1177	card: 950	utexas: 713	math: 666	stereo: 655	true: 832
data: 1044	serial: 748	dx: 713	coprocessor: 639	channel: 588	slot: 778
sec: 1044	ports: 748	cc: 688	writes: 586	play: 565	quadra: 671
dma: 1025	work: 662	ultra: 586	cd: 559	problems: 542	lc: 671

4 vs. 3 task					
topics for comp.sys.ibm.pc.hardware 4			topics for comp.os.ms-windows.misc 3		
4.0	3.8	2.5	4.2	2.5	2.3
bus: 4120	work: 1933	monitor: 2339	windows: 10119	article: 3449	text: 1119
isa: 2472	writes: 1933	question: 1595	win: 2413	find: 2362	problem: 1085
data: 1960	dos: 1750	thing: 1595	nt: 2231	change: 1650	software: 984
dma: 1292	boot: 1463	apr: 992	ms: 1278	editor: 1125	gateway: 950
eisa: 1225	sys: 1202	kind: 922	os: 1278	internet: 825	shareware: 916
os: 1181	article: 993	heard: 922	programs: 710	wrote: 750	hardware: 678
system: 1158	apr: 706	robert: 780	groups: 527	rob: 638	cursor: 678
vlb: 1158	config: 706	mine: 780	server: 507	changing: 600	recently: 645
meg: 958	wrote: 653	monitors: 780	create: 487	previous: 563	writes: 611
don: 824	adaptec: 601	vga: 638	megs: 385	csd: 563	applications: 611

Shared topics for 4 vs. 5			Shared topics for 4 vs. 3		
4.1	3.5	3.3	5.5	3.4	3.2
simms: 2750	monitor: 2170	writes: 2719	card: 4220	port: 2268	scsi: 8024
ns: 1528	good: 1511	article: 2504	drivers: 2063	irq: 2192	drive: 5155
simm: 1032	uk: 1043	cd: 2269	driver: 2027	modem: 1601	hard: 3571
meg: 1012	ac: 894	time: 1839	video: 1719	serial: 1449	drives: 3063
mail: 955	nec: 724	access: 1311	screen: 1316	pc: 1334	controller: 3063
data: 841	quality: 639	problems: 117	cards: 889	mouse: 1239	floppy: 1868
chips: 688	article: 575	de: 998	ati: 889	board: 1182	supports: 986
pin: 554	buy: 532	rom: 978	diamond: 735	problems: 1087	standard: 762
writes: 497	box: 511	ms: 783	version: 688	work: 1067	problem: 598
cwru: 459	price: 490	toshiba: 626	local: 688	nec: 934	mfm: 598

Table 4.6: Comparison of individual topics learned discriminatively for two binary tasks. Each column within a table lists the ten most popular words for an individual topic ϕ_k^y . The first two tables list “strongly” class-dependent topics. The last table contrasts the “strongly” shared topics for each task. The number above each list of words represents the strength of the topic, as described in the text.

Chapter 4. DiscLDA: Discriminative Dimensionality Reduction for Classification

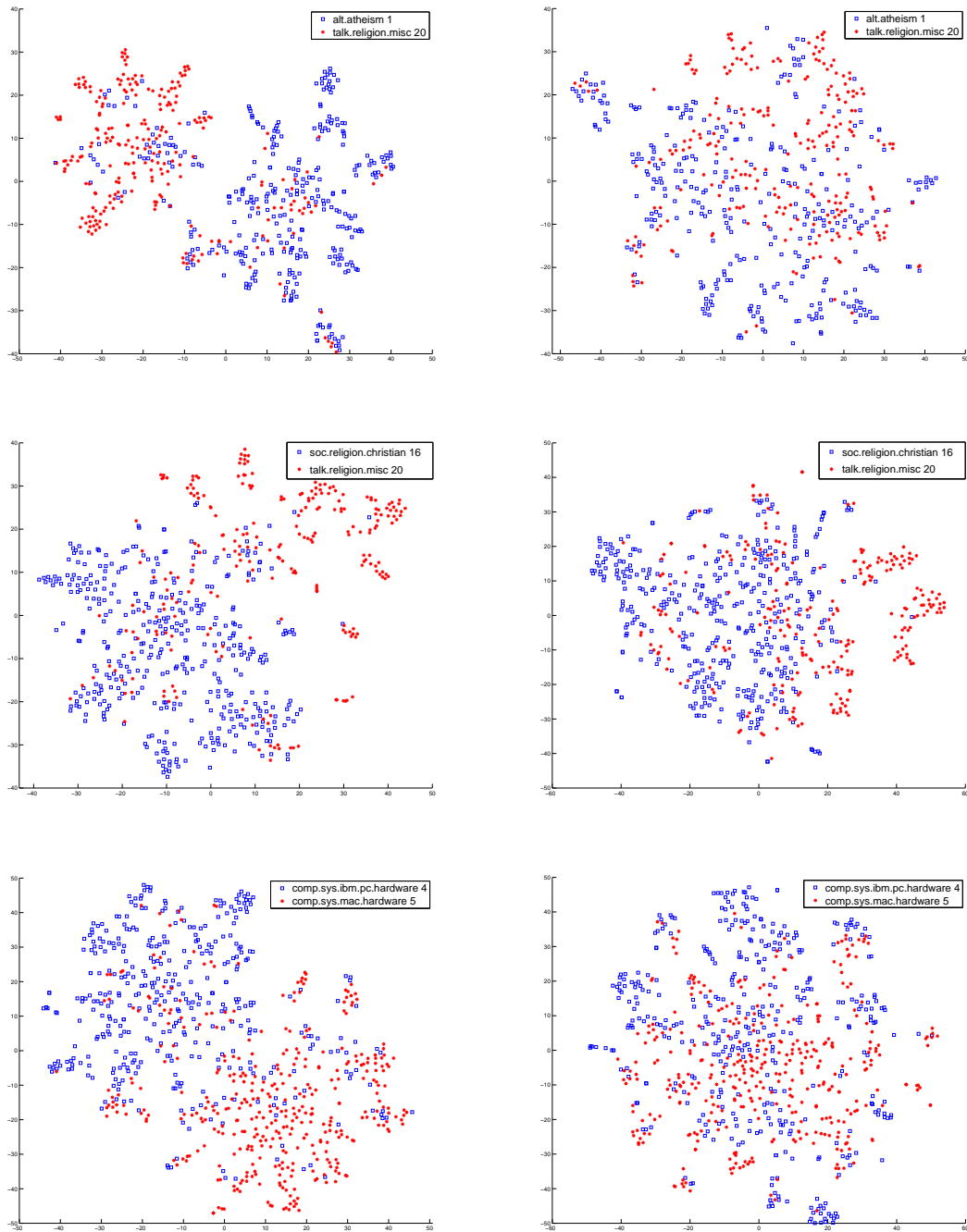


Figure 4.8: t-SNE 2D embedding of the DiscLDA (left) vs. LDA (right) representation of documents for several binary tasks, where T^y is learned discriminatively this time for DiscLDA. From top to bottom: 1 vs. 20, 16 vs. 20 and 4 vs. 5.

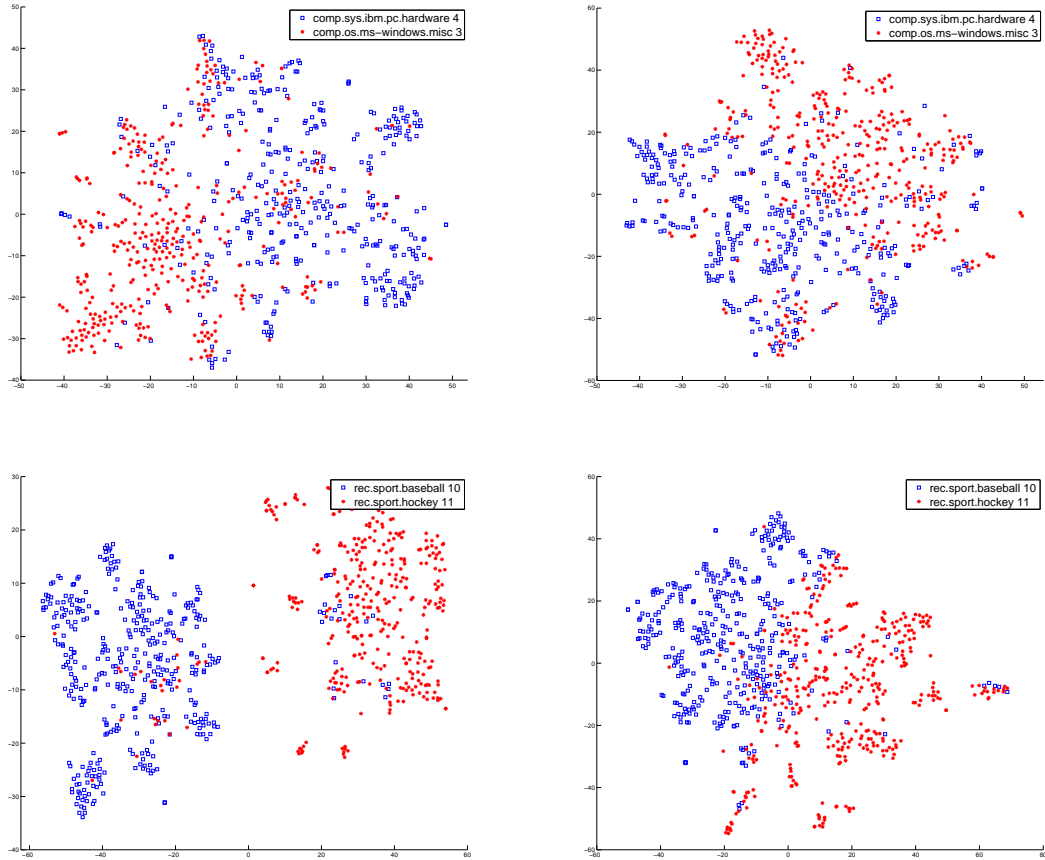


Figure 4.9: t-SNE 2D embedding of the DiscLDA (left) vs. LDA (right) representation of documents for the remaining binary tasks (continuation of Fig. 4.8). From top to bottom: 4 vs. 3 and 10 vs. 11.

4.5 Discussion

4.5.1 Related work

4.5.1.1 Author-topic model

By including a Dirichlet prior on the \mathbf{T}^y parameters, the DiscLDA model can be related to the author-topic model of [Rosen-Zvi et al. \(2004\)](#), in the special case in which there is only one author per document, as shown in [Fig. 4.10](#) (using our notation for making the link more explicit). In the author-topic model, the bag-of-words representation of a document is augmented by a list of the authors of the document. To generate a word in a document, one first picks at random the author associated with this document. Given the author (y in our notation), a topic is chosen according to *corpus-wide* author-specific topic-mixture proportions (which is a column vector \mathbf{T}_y in our notation). The word is then generated from the corresponding topic distribution as usual. According to this analogy, we see that our model not only enables us to predict the author of a document (assuming a small set of possible authors), but we also capture the content of documents (using $\boldsymbol{\theta}$) as well as the corpus-wide class properties (using \mathbf{T}). The focus of the author-topic model was to model the interests of authors, not the content of documents, explaining why there was no need to add document-specific topic-mixture proportions. Because we want to predict the class for a *specific* document, it is crucial in our case that we also model the content of a document.

4.5.1.2 Supervised topic models

Recently, there has been growing interest in topic modeling with supervised information. [Blei and McAuliffe \(2008\)](#) proposed a supervised LDA model where the empirical topic vector \mathbf{z} (sampled from $\boldsymbol{\theta}$) is used as a covariate for a regression on y

— see also [Flaherty et al. \(2005\)](#). Their graphical model is shown in Fig. 4.12. The generative process is the same as in LDA, but in addition, the output variable y_d is distributed as a Generalized Linear Model (GLM) with the empirical counts of z_d as input. [Mimno and McCallum \(2008\)](#) proposed the Dirichlet-multinomial regression model which can handle various types of side information, including the case in which this side information is an indicator variable of the class (y)¹¹. We show a translation of their graphical model to our notation in Fig. 4.13. Our work differs from theirs, however, in that we train the transformation parameter by maximum conditional likelihood instead of a generative criterion.

On the other hand, [Zhu et al. \(2009\)](#) proposed a max-margin variant of the supervised LDA model of [Blei and McAuliffe \(2008\)](#) by using the Maximum Entropy Discrimination framework of [Jaakkola et al. \(1999\)](#). This is the only other discriminatively trained variation of topic models that we are aware of, and probably the closest in spirit to our work. Another relevant piece of work which combines a generative and discriminative training criterion for a latent variable model for text classification is the work from [Larochelle and Bengio \(2008\)](#). In their case, they build on the recent progress on Restricted Boltzmann Machines to obtain a latent variable model of documents which also yields an impressive prediction accuracy (and unlike ours, is quite competitive to pure classifiers).

4.5.2 Caveats and future directions

Our experiments with DiscLDA presented encouraging results in its ability to obtain a reduced representation of documents which could be interpretable while still maintaining its predictive performance. On the other hand, we have several caveats to make. First of all, even though moving from the harmonic mean estimator to

¹¹In this case, their model is actually the same as Model 1 in [Fei-fei and Perona \(2005\)](#) with an additional prior on the class-dependent parameters for the Dirichlet distribution on the topics.

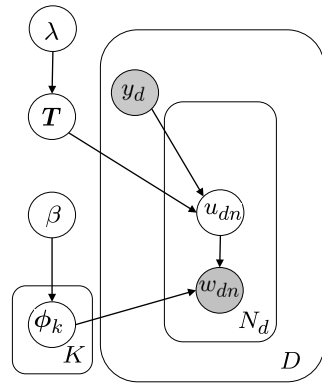


Figure 4.10: Author-topic model with only one author y_d per document and using our notation.

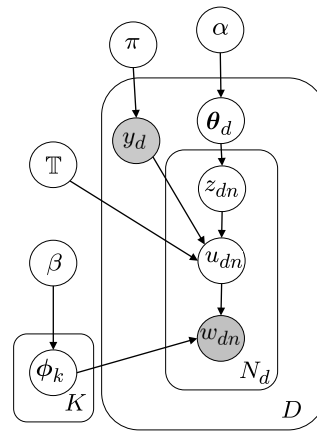


Figure 4.11: DiscLDA model again for comparison.

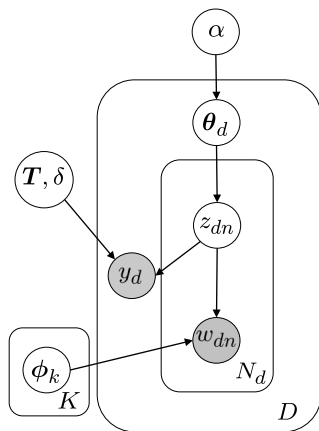


Figure 4.12: Supervised LDA model.

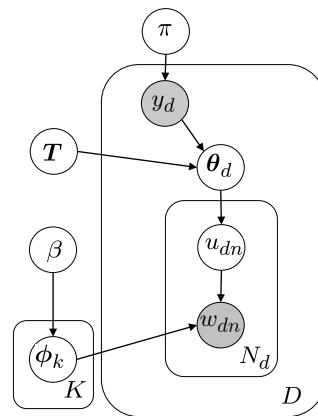


Figure 4.13: A special case of the Dirichlet-multinomial regression model.

the bridge sampling one decreased the noise of the class-posterior estimates (which are used ubiquitously in our algorithm: dimensionality reduction, estimating the gradient, prediction, etc.), it stayed significant enough to leave the validation error with wide fluctuations during the algorithm. It now appears to us that a variational approach would probably be much more appropriate to the discriminative learning framework, as the noise in the various stochastic part of the sampling approach is compounded rather than averaged out. Moreover, due to time constraints, we haven't done a stability analysis of our results. A more in depth analysis would include different initialization points, deviation measures for different samples and other sources of variation, etc. Given the recent work by [Asuncion et al. \(2009\)](#) which showed that the performance of LDA crucially depends on the setting of its hyperparameters, we also think that α and β should be also tuned, especially with a classification task in mind.

One aspect of the DiscLDA probabilistic model that we haven't presented in this work is to consider a Bayesian average over Φ for prediction. In some exploratory experiments, we actually observed a significant accuracy increase for a version of the predictive DiscLDA model which classifies using $\sum_i p(y|\mathbf{w}_d, \mathbb{T}, \Phi^{(i)})$ with several point estimates $\Phi^{(i)} = [\Phi|\mathbf{u}^{(i)}, \mathbf{w}]$ rather than just using one sample as in step 2 of the algorithm of Table 4.1. The problem is that we didn't figure out how to interleave the discriminative update on \mathbf{T}^y with those several point estimates, even though it would seem like a good thing to do. Another subtlety comes from the non-identifiability of Φ because of the symmetry of its prior: it would yield a moving target for the \mathbf{T}^y updates, which would be affected by Φ jumping around different permutations of indices. One way to solve this problem would be to use an *asymmetric* prior on Φ (or on θ) – which we haven't seen much explored yet in the topic modeling literature – but we think is a promising venue. A meaningful prior that we have in mind would be for example to use a geometrically decreasing series of the α_l parameters: this would

induce an ordering of the topics in term of their importance in the corpus, and would thus break the symmetry. The columns of Φ would then become identifiable and we could combine multiple samples to estimate it, for example; or combine multiple $\Phi^{(i)}$ to estimate the new representation of a documents $\sum_i \frac{1}{M} \mathbb{E} \left[\mathbf{T}^y \boldsymbol{\theta}_d | \mathbf{w}_d, \mathbb{T}, \Phi^{(i)} \right]$, which would be more stable. We leave this as an interesting avenue for future work.

4.6 Summary

We have presented DiscLDA, a variation on LDA in which the LDA parametrization is augmented to include a transformation matrix and in which this matrix is learned via a conditional likelihood criterion. This approach allows DiscLDA to retain the ability of the LDA approach to find useful low-dimensional representations of documents, but to also make use of discriminative side information (labels) in forming these representations.

Although we have focused on LDA, we view our strategy as more broadly useful. A virtue of the probabilistic modeling framework is that it can yield complex models that are modular and can be trained effectively with unsupervised methods. Given the high dimensionality of such models, it may be intractable to train all of the parameters via a discriminative criterion such as conditional likelihood. In this case it may be desirable to pursue a mixed strategy in which we retain the unsupervised criterion for the full parameter space but augment the model with a carefully chosen transformation so as to obtain an auxiliary low-dimensional optimization problem for which conditional likelihood may be more effective.

Appendix 4.A Derivations

4.A.1 Gradient equations

In this section, we show how to derive the relevant quantities for estimating the gradient in equation (4.7) of the algorithm in Table 4.1.

4.A.1.1 Using an EM bound

We first present how to compute the derivative of the marginal likelihood of a function using the expectation of the derivative of the complete likelihood. This property is often overlooked in the presentation of the EM algorithm, so we rederive here for convenience. We switch temporarily to generic notation for this presentation. Assume you have a probability distribution $p(x, z|\theta)$ where z is a hidden variable. We want to compute $\frac{\partial}{\partial\theta} \log p(x|\theta)$. Using Jensen's inequality, we have:

$$L(\theta) = \log p(x|\theta) \geq \int_z q(z) \log \left(\frac{p(x, z|\theta)}{q(z)} \right) \doteq f_q(\theta)$$

for any fixed distribution $q(z)$. Moreover,

$$L(\theta) - f_q(\theta) = \text{KL}(q(z) \parallel p(z|x, \theta))$$

and so we have

$$L(\theta_t) = f_{q_t}(\theta_t)$$

for $q_t(z) = p(z|x, \theta_t)$ (for a fixed θ_t). Since we have a differentiable lower bound of a function which agrees at one point, we know that their tangent agrees at this point – one way to see this is that the derivative of $L(\theta) - f_q(\theta)$ is zero at θ_t because it is a minimum. Under suitable regularity conditions, we can push the derivative of $f_{q_t}(\theta)$ with respect to θ inside the integral to obtain that $\frac{\partial}{\partial\theta} f_{q_t}(\theta) = \mathbb{E}_{q_t} \left[\frac{\partial}{\partial\theta} \log p(x, z|\theta) \right]$,

Hence, we have in general the following gradient matching property:

$$\frac{\partial}{\partial \theta} \log p(x|\theta) \Big|_{\theta=\theta_t} = \mathbb{E}_{q_t} \left[\frac{\partial}{\partial \theta} \log p(x, z|\theta) \Big|_{\theta=\theta_t} \right] \quad (4.12)$$

for $q_t(z) \doteq p(z|x, \theta_t)$.

4.A.1.2 Gradient derivation

In our case, we want to compute: $\frac{\partial}{\partial T_{kl}^y} \log p(\mathbf{y}|\mathbf{w}, \mathbb{T}, \Phi) = \sum_d \frac{\partial}{\partial T_{kl}^y} \log p(y_d|\mathbf{w}_d, \mathbb{T}, \Phi)$, as conditioning on Φ separates the components nicely over documents. The log-posterior over labels can be written as:

$$\log p(y_d|\mathbf{w}_d, \mathbb{T}, \Phi) = \log p(y_d) + \log p(\mathbf{w}_d|y_d, \mathbb{T}, \Phi) - \log p(\mathbf{w}_d|\mathbb{T}, \Phi)$$

and so

$$\frac{\partial}{\partial T_{kl}^y} \log p(y_d|\mathbf{w}_d, \mathbb{T}, \Phi) = \frac{\partial}{\partial T_{kl}^y} \log p(\mathbf{w}_d|y_d, \mathbb{T}, \Phi) - \frac{\partial}{\partial T_{kl}^y} \log p(\mathbf{w}_d|\mathbb{T}, \Phi) \quad (4.13)$$

as the prior $p(y_d)$ doesn't depend on \mathbb{T} . We can use the gradient matching property from equation (4.12) to compute each of the terms of the RHS of (4.13) using an expectation over a posterior over \mathbf{z}_d . For example:

$$\frac{\partial}{\partial T_{kl}^y} \log p(\mathbf{w}_d|\mathbb{T}, \Phi) \Big|_{\mathbb{T}=\mathbb{T}_t} = \mathbb{E}_{r_t^d(\mathbf{z}_d)} \left[\frac{\partial}{\partial T_{kl}^y} \log p(\mathbf{w}_d, \mathbf{z}_d|\mathbb{T}, \Phi) \Big|_{\mathbb{T}=\mathbb{T}_t} \right] \quad (4.14)$$

for $r_t^d(\mathbf{z}_d) \doteq p(\mathbf{z}_d|\mathbf{w}_d, \mathbb{T}_t, \Phi)$.

We can obtain the first term of the RHS of equation (4.13) similarly using $q_t^{y_d}(\mathbf{z}_d) \doteq p(\mathbf{z}_d|y_d, \mathbf{w}_d, \mathbb{T}_t, \Phi)$, and so we obtain a document-level version of the gradient matching equation (4.8). We can approximate those integrals using Gibbs samples from the posteriors $q_t^{y_d}(\mathbf{z}_d)$ and $r_t^d(\mathbf{z}_d)$. However, as we mentioned in Sec. 4.3.1, the chain

for $r_t^d(\mathbf{z}_d)$ doesn't mix well, hence we instead integrate out y_d analytically in the expectation of (4.14) using the tower property of conditional expectations:

$$\mathbb{E}[g(\mathbf{z}_d) | \mathbf{w}_d, \mathbb{T}_t, \Phi] = \mathbb{E}[\mathbb{E}[g(\mathbf{z}_d) | y_d, \mathbf{w}_d, \mathbb{T}_t, \Phi] | \mathbf{w}_d, \mathbb{T}_t, \Phi],$$

where we defined $g(\mathbf{z}_d) = \frac{\partial}{\partial T_{kl}^y} \log p(\mathbf{w}_d, \mathbf{z}_d | \mathbb{T}, \Phi)$. Equation (4.14) thus becomes:

$$\left. \frac{\partial}{\partial T_{kl}^y} \log p(\mathbf{w}_d | \mathbb{T}, \Phi) \right|_{\mathbb{T}=\mathbb{T}_t} = \sum_{y'} p(y' | \mathbf{w}_d, \mathbb{T}_t, \Phi) \mathbb{E}_{q_t^{y'}(\mathbf{z}_d)} \left[\left. \frac{\partial}{\partial T_{kl}^y} \log p(\mathbf{w}_d, \mathbf{z}_d | \mathbb{T}, \Phi) \right|_{\mathbb{T}=\mathbb{T}_t} \right]. \quad (4.15)$$

We use bridge sampling to estimate $p(y' | \mathbf{w}_d, \mathbb{T}_t, \Phi)$ and Monte-Carlo integration with Gibbs samples from $q_t^{y'}(\mathbf{z}_d)$ to estimate the expectations in (4.15).

We now derive the expressions for the required derivatives. From

$$p(\mathbf{w}_d, \mathbf{z}_d | \mathbb{T}, \Phi) = p(\mathbf{z}_d) \sum_{y'} p(y') p(\mathbf{w}_d | y', \mathbf{z}_d, \mathbb{T}, \Phi),$$

and since $\frac{\partial}{\partial T_{kl}^y} p(\mathbf{w}_d | y', \mathbf{z}_d, \mathbb{T}, \Phi) = 0$ for $y' \neq y$, we get

$$\begin{aligned} \frac{\partial}{\partial T_{kl}^y} \log p(\mathbf{w}_d, \mathbf{z}_d | \mathbb{T}, \Phi) &= \frac{p(\mathbf{z}_d) p(y)}{p(\mathbf{w}_d, \mathbf{z}_d | \mathbb{T}, \Phi)} \frac{\partial}{\partial T_{kl}^y} \exp(\log p(\mathbf{w}_d | y, \mathbf{z}_d, \mathbb{T}, \Phi)) \\ &= \frac{p(\mathbf{z}_d) p(y) p(\mathbf{w}_d | y, \mathbf{z}_d, \mathbb{T}, \Phi)}{p(\mathbf{w}_d, \mathbf{z}_d | \mathbb{T}, \Phi)} \frac{\partial}{\partial T_{kl}^y} \log p(\mathbf{w}_d | y, \mathbf{z}_d, \mathbb{T}, \Phi) \\ &= p(y | \mathbf{z}_d, \mathbf{w}_d, \mathbb{T}, \Phi) \frac{\partial}{\partial T_{kl}^y} \log p(\mathbf{w}_d | y, \mathbf{z}_d, \mathbb{T}, \Phi). \end{aligned}$$

The conditional of y given \mathbf{z}_d is easily obtained:

$$p(y | \mathbf{z}_d, \mathbf{w}_d, \mathbb{T}, \Phi) \propto \pi_y \prod_n \Phi_{\mathbf{w}_{dn}, \mathbf{z}_{dn}}^y, \quad (4.16)$$

where we have defined the label-dependent topic parameters – as we did in equa-

tion (4.3) – as

$$\Phi_{vl}^y \doteq \sum_k \Phi_{vk} T_{kl}^y.$$

With this notation, we have

$$\frac{\partial}{\partial T_{kl}^y} \log p(\mathbf{w}_d | y_d, \mathbf{z}_d, \mathbb{T}, \Phi) = \sum_n \frac{\Phi_{w_{dn}k}}{\Phi_{w_{dn}l}^y} \delta(l, z_{dn}) \delta(y, y_d).$$

where $\delta(k, l)$ is 1 when $k = l$ and 0 otherwise.

Combining these equations together and using a Monte-Carlo estimate for the approximations of the expectations, we get the following final estimate for the gradient:

$$\begin{aligned} \frac{\partial}{\partial T_{kl}^y} \log p(y_d | \mathbf{w}_d, \mathbb{T}, \Phi) \approx & \quad (4.17) \\ & \sum_{y'} \frac{1}{M} \sum_i \left[\delta(y', y_d) \delta(y, y_d) - \hat{p}(y' | \mathbf{w}_d, \mathbb{T}, \Phi) p(y | \mathbf{z}_d^{(i)y'}, \mathbf{w}_d, \mathbb{T}, \Phi) \right] \\ & \left(\sum_n \frac{\Phi_{w_{dn}k}}{\Phi_{w_{dn}l}^y} \delta(l, z_{dn}^{(i)y'}) \right), \end{aligned}$$

where $\mathbf{z}_d^{(i)y'}$ is sampled for $i = 1, \dots, M$ from $p(\mathbf{z}_d | y', \mathbf{w}_d, \mathbb{T}, \Phi)$ for $y' = 1, \dots, C$; $\hat{p}(y' | \mathbf{w}_d, \mathbb{T}, \Phi)$ is estimated using bridge sampling – see equation (4.33)¹²; and finally $p(y | \mathbf{z}_d^{(i)y'}, \mathbf{w}_d, \mathbb{T}, \Phi)$ is given in equation (4.16).

Finally, we reparameterize T_{kl}^y with t_{kl}^y in the log-domain using a soft-max transformation so that it stays automatically normalized:

$$T_{kl}^y \doteq \frac{\exp(t_{kl}^y)}{\sum_r \exp(t_{rl}^y)}.$$

¹²We could reuse the same $\mathbf{z}_d^{(i)y'}$ samples in the bridge sampling estimate to reduce the variance, though this was not done in our implementation. On the other hand, note that the formulation (4.17) reused the same $\mathbf{z}_d^{(i)y'}$ samples to estimate the two terms of equation (4.8)

The derivative for this new parameterization is given simply via the chain rule as:

$$\frac{\partial}{\partial t_{kl}^y} = T_{kl}^y(1 - T_{kl}^y) \frac{\partial}{\partial T_{kl}^y}.$$

4.A.2 Gibbs sampling equations

We start by introducing the notation for this section. $\alpha_{(\cdot)}$ means the summation over its index, i.e. $\alpha_{(\cdot)} \doteq \sum_{l=1}^L \alpha_l$. The discrete (multinomial of size 1) variables take values in $\{1, \dots, \text{maxrange}\}$ unless *an extra index* is added to them (and the context tells whether an extra index was added; sometimes the name of the letter used as an index is unfortunately meaningful). E.g. $z_{dn} \in \{1, \dots, L\}$ but $z_{dnk} \in \{0, 1\}$; and \mathbf{z}_d is a vector of discrete variables. $a^{[x]}$ (with the brackets in the exponent instead of parenthesis) is the rising factorial:

$$a^{[n]} \doteq a(a+1) \cdots (a+n-1).$$

We provide the equations in this section for the DiscLDA formulation in Fig. 4.3 with general Dirichlet priors with hyperparameters $\{\alpha_l\}_{l=1}^L$ and $\{\beta_{vk}\}_{v=1}^V$ for $k = 1, \dots, K$ even though we use symmetric priors in the experimental section. Finally, we omit the hyperparameters in the conditionals, even though they are implicitly understood.

To get the Rao-Blackwellized updates, we need the marginals of our generative model. Marginalizing $\boldsymbol{\theta}_d$ out, we get:

$$p(\mathbf{z}_d) = \frac{\prod_{l=1}^L \alpha_l^{[n_l^{(d)}]}}{\alpha^{[N_d]}} \quad \text{where } n_l^{(d)} \doteq \sum_{n=1}^{N_d} z_{dnl}. \quad (4.18)$$

Marginalizing Φ out, we get:

$$p(\mathbf{w}|\mathbf{u}) = \prod_{k=1}^K \left(\frac{\prod_{v=1}^V \beta_{vk}^{[m_k^{(v)}]}}{\beta_{(\cdot)k}^{[m_k^{(\cdot)}]}} \right) \quad \text{where } m_k^{(v)} \doteq \sum_{d=1}^D \sum_{n=1}^{N_d} u_{dnk} w_{dnv}. \quad (4.19)$$

The count $n_l^{(d)}$ thus represents how often the topic l was chosen in the document d . On the other hand, the count $m_k^{(v)}$ is *corpus wide* and counts the number of time word v was assigned to topic k in the whole corpus.

Marginalizing u_{dn} out, we get the effect of the transformation \mathbf{T}^{y_d} as a class-dependent mixture of the word distribution for topic z_{dn} :

$$p(w_{dn}|z_{dn}, y_d, \Phi) = \sum_{k=1}^K \Phi_{w_{dn}k} T_{kz_{dn}}^{y_d} \doteq \Phi_{w_{dn}z_{dn}}^{y_d}. \quad (4.20)$$

Marginalizing z_{dn} out (but keeping θ_d), we get another view of the effect of the transformation (on the topic mixture this time instead of the word distribution):

$$p(u_{dn}|\theta_d, y_d, \mathbb{T}) = \sum_{l=1}^L T_{u_{dn}l}^{y_d} \theta_{dl}. \quad (4.21)$$

It is not possible to marginalize θ_d out to get $p(u_{dn}|y_d, \mathbb{T})$ in closed form though as θ_d doesn't have a conjugate prior to the distribution on u_{dn} . As in LDA, $p(u_{dn}|y_d, \mathbb{T})$ can be expressed in terms of hypergeometric functions (see [Carlson, 1977](#), e.g.), but the expression obtained is not simple to deal with.

Marginalizing introduces dependencies on the siblings of the variables; so we introduce another piece of notation: a bar on an index means that we consider all the variables with a different value for this index (we are taking the set complement). We now provide the equations needed to implement the RB Gibbs updates used in this chapter. To sample from $p(\mathbf{z}|\mathbf{y}, \mathbf{w}, \mathbb{T}, \Phi)$ (which decouples in independent updates

for each document d), we need:

$$p(z_{dn} = l | y_d, \mathbf{z}_{d\bar{n}}, \mathbf{w}_d, \mathbb{T}, \Phi) \propto \Phi_{w_{dn}l}^{y_d} (\alpha_l + n_{\bar{n},l}^{(d)}) \quad (4.22)$$

where

$$n_{\bar{n},l}^{(d)} \doteq \sum_{\substack{i=1 \\ i \neq n}}^{N_d} z_{dil}. \quad (4.23)$$

To sample from $p(\mathbf{u} | \mathbf{z}, \mathbf{y}, \mathbf{w}, \mathbb{T}, \Phi)$, we need:

$$p(u_{dn} = k | z_{dn}, y_d, w_{dn}, \mathbb{T}, \Phi) \propto \Phi_{w_{dn}k} T_{k z_{dn}}^{y_d}. \quad (4.24)$$

Finally, to sample from $p(\mathbf{u}, \mathbf{z} | \mathbf{y}, \mathbf{w}, \mathbb{T})$ with Φ marginalized out, we need:

$$p(u_{dn} = k, z_{dn} = l | \mathbf{u}_{d\bar{n}}, \mathbf{z}_{d\bar{n}}, y_d, \mathbf{w}, \mathbb{T}) \propto \frac{(\beta_{w_{dn}k} + m_{\bar{d}n,k}^{(w_{dn})})}{(\beta_{(\cdot)k} + m_{\bar{d}n,k}^{(\cdot)})} T_{kl}^{y_d} (\alpha_l + n_{\bar{n},l}^{(d)}) \quad (4.25)$$

which doesn't factorize as independent updates over d , and where we have defined:

$$m_{\bar{d}n,k}^{(v)} \doteq \sum_{i=1}^D \sum_{\substack{j=1 \\ (i,j) \neq (d,n)}}^{N_i} u_{ijk} \delta(v, w_{ij}). \quad (4.26)$$

Note that computing the joint scales quadratically in terms of topic size. This makes sampling in this case K times slower than Gibbs sampling in LDA. A faster alternative would be to sample u_{dn} and z_{dn} one after each other rather than jointly. Unfortunately, because the \mathbf{T}^y matrices are usually close to be sparse, the chain wouldn't mix well as given z_{dn} , u_{dn} would become almost deterministic. It is thus crucial to sample them jointly in order to allow exploration. Moreover, because the updates depend on the whole corpus through \mathbf{w} , one needs to cache (u_{dn}, z_{dn}) for all tokens, an expensive operation. This is an obstacle to scale to a large number of topics without

some modifications to the approximation (e.g., we could use the Metropolis-Hasting algorithm with a suitable proposal distribution rather than Gibbs sampling).

The posterior over topic parameters (which factorizes over k) is:

$$p(\boldsymbol{\phi}_k | \mathbf{u}, \mathbf{w}) = \text{Dir} \left(\{\beta_{vk} + m_k^{(v)}\}_{v=1}^V \right), \quad (4.27)$$

so the posterior mean is simply the normalized Dirichlet parameters.

The posterior over topic proportions (which factorizes over d) is:

$$p(\boldsymbol{\theta}_d | \mathbf{z}_d) = \text{Dir} \left(\{\alpha_l + n_l^{(d)}\}_{l=1}^L \right). \quad (4.28)$$

4.A.3 Dimensionality reduction equation

Using equation (4.10), we can expand equation (4.9) as:

$$\left(\mathbb{E} [\mathbf{T}^y \boldsymbol{\theta}_d | \mathbf{w}_d, \mathbb{T}, \boldsymbol{\Phi}] \right)_k \approx \sum_y \hat{p}(y | \mathbf{w}_d, \mathbb{T}, \boldsymbol{\Phi}) \sum_l T_{kl}^y \frac{\alpha_l + \hat{n}_l^{(d)}(y)}{\alpha_{(\cdot)} + N_d}, \quad (4.29)$$

where $\hat{p}(y | \mathbf{w}_d, \mathbb{T}, \boldsymbol{\Phi})$ is estimated as before using bridge sampling, and

$$\hat{n}_l^{(d)}(y) \doteq \frac{1}{M} \sum_{i,n} \delta(l, z_{dn}^{(i)y})$$

for $\mathbf{z}_d^{(i)y}$ sampled for $i = 1, \dots, M$ from $p(\mathbf{z}_d | y, \mathbf{w}_d, \mathbb{T}, \boldsymbol{\Phi})$.

Appendix 4.B Bridge sampling

Bridge sampling (Meng and Wong, 1996) can be seen as an extension of importance sampling to estimate the ratio between two normalization factors. We review the approach in this section and apply it to our setting to estimate the posterior over

labels $p(y|\mathbf{w}_d, \mathbb{T}, \Phi)$. We note that our presentation is slightly different than the one in the literature.

We will first motivate the approach as an improvement over importance sampling. Suppose one is interested in computing the integral $\mathbb{E}_1[h] \doteq \int h(z)p_1(z)dz$ ¹³ for some density $p_1 = q_1/Z_1$, where Z_1 is its normalization constant. Importance sampling approximates the integral using samples from a different proposal density $p_0 = q_0/Z_0$:

$$\mathbb{E}_1[h] = \frac{\int \frac{q_1}{q_0} h p_0}{\frac{Z_1}{Z_0}} = \frac{\mathbb{E}_0 \left[\frac{q_1 h}{q_0} \right]}{\mathbb{E}_0 \left[\frac{q_1}{q_0} \right]} \approx \frac{\frac{1}{M} \sum_{i=1}^M w(z^{(i)}) h(z^{(i)})}{\frac{1}{M} \sum_{i=1}^M w(z^{(i)})}, \quad (4.30)$$

where we have defined the *importance weights* $w(z^{(i)}) \doteq \frac{q_1(z^{(i)})}{q_0(z^{(i)})}$ and we used the standard Monte Carlo approximation for the expectation \mathbb{E}_0 with M samples from the proposal p_0 i.e. $z^{(i)} \sim p_0(z)$. The same samples were used for the numerator and denominator to reduce the sampling error of the estimate. This formula is valid for any proposal density p_0 which has a wider support than the original density, that is, $\Omega_0 \supseteq \Omega_1$ where $\Omega_c \doteq \{z : p_c(z) > 0\}$ for $c \in \{0, 1\}$. Note that the only two requirements to be able to evaluate the approximation are 1) to be able to evaluate the *unnormalized* distributions q_0, q_1 and 2) to be able to sample from the proposal p_0 – hence we don’t need to know the normalized p_0, p_1 , which is a powerful advantage of the method.

Importance sampling is normally used within two scenarios. In the first one, we are not able to sample from p_1 and instead decide to sample from a suitably close proposal distribution q_0 . In the second scenario, we are able to sample from p_1 , but the

¹³ The index for \mathbb{E} will indicate which distribution the expectation is taken with respect to (p_1 or p_0 e.g.). For more generality, we could replace dz in the integral with $\mu(dz)$ and assume that all densities are taken with respect to the base measure μ . For simplicity, we will only consider the Lebesgue measure in this presentation, though the derivations carry through with other measures (e.g. a mix of counting and Lebesgue measure as is needed for the LDA model which has discrete and continuous variables).

naive Monte Carlo estimate of $\mathbb{E}_1[h]$ converges poorly because the major contributions of h are far from where the major mass of p_1 is. Importance sampling allows us to focus on the *important* regions of the integral by using a proposal q_0 better calibrated to h , e.g., close to $q_1 h$, and thus yielding a more efficient estimator. See chapters 2-4 of [Liu \(2001\)](#) for a more extended discussion of importance sampling in the context of Markov chain simulation algorithms.

In order to link this approach to evaluating the ratio of normalization constants, recall that the denominator in (4.30) is using the identity:

$$\frac{Z_1}{Z_0} = \mathbb{E}_0 \left[\frac{q_1}{q_0} \right] \approx \frac{1}{M} \sum_{i=1}^M w(z^{(i)}), \quad (4.31)$$

which provides an approximation to the ratio Z_1/Z_0 . A problem arises when q_0 is quite far to q_1 and in particular, if it has thinner tails than q_1 . In this case, some low probability z 's under q_0 will have high weights $\frac{q_1(z)}{q_0(z)}$ and the variance of the estimator will be high — and could actually be infinite in certain examples as in the harmonic mean estimator for the marginal likelihood ([Newton and Raftery, 1994](#)). A way to solve the problem is to use a *bridge distribution* q_b between q_0 and q_1 :

$$\frac{\mathbb{E}_0 \left[\frac{q_b}{q_0} \right]}{\mathbb{E}_1 \left[\frac{q_b}{q_1} \right]} = \frac{Z_b/Z_0}{Z_b/Z_1} = \frac{Z_1}{Z_0},$$

which is valid for any bridge distribution q_b which is normalizable and such that $\Omega_b \subseteq \Omega_0 \cap \Omega_1$. Using Monte Carlo approximations for both the numerator and denominator, we get:

$$\frac{Z_1}{Z_0} \approx \frac{\frac{1}{M_0} \sum_{i=1}^{M_0} \frac{q_b}{q_0}(z_0^{(i)})}{\frac{1}{M_1} \sum_{i=1}^{M_1} \frac{q_b}{q_1}(z_1^{(i)})} \quad (4.32)$$

where the $z_c^{(i)}$'s are M_c iid samples from the distribution q_c for $c \in \{0, 1\}$. This is

called the *bridge sampling* estimate of Z_1/Z_0 using the *bridge distribution* q_b . q_b acts as a 'bridge' between q_0 and q_1 by reducing the distribution discrepancy for q_b/q_0 and q_b/q_1 vs. q_1/q_0 . More specifically, one can choose q_b to have thinner tails than both q_1 and q_0 , hence avoiding the possibility of extremely high weights with low probability (see Fig. 4.14 for an example).

Meng and Wong (1996) show that the choice of bridge distribution q_b which minimizes the asymptotic relative mean-square error of the bridge sampling estimate of Z_1/Z_0 (assuming $M_0 = M_1$) is the harmonic mean of p_0 and p_1 : $q_b = (\frac{Z_0}{q_0} + \frac{Z_1}{q_1})^{-1}$. Unfortunately, the ratio Z_1/Z_0 needs to be known to compute this optimal distribution. They thus propose an iterative scheme (based on a fixed point equation) which yields an estimator which doesn't require the ratio Z_1/Z_0 but still has the same asymptotic relative mean square error as the harmonic mean bridge. We refer the readers to their paper for more details. In our limited experiments with DiscLDA (see Sec. 4.B.1), we found that the iterative scheme didn't improve much on the simpler *geometric bridge* $q_b = \sqrt{q_0 q_1}$, which has much lower computational complexity.

Link with importance sampling Note that using the bridge $q_b = q_1$ makes the denominator in (4.32) disappear and yields back the importance sampling estimate for the ratio Z_1/Z_0 as given in (4.31).

Likelihood ratio application We can use bridge sampling to compute the ratio of marginal likelihoods. Consider $q_c(z) \doteq p(x, z|y_c)$; then $Z_c = p(x|y_c)$ and thus Z_1/Z_0 is the likelihood ratio between class y_1 and class y_0 .

Other links Bridge sampling can be used to estimate a marginal likelihood $Z_1 \doteq \int_z p(x, z)$ directly by letting $q_1(z) \doteq p(x, z)$ and by using a proposal distribution q_0 which is already normalized (and so $Z_0 = 1$) – e.g., one could use a Gaussian

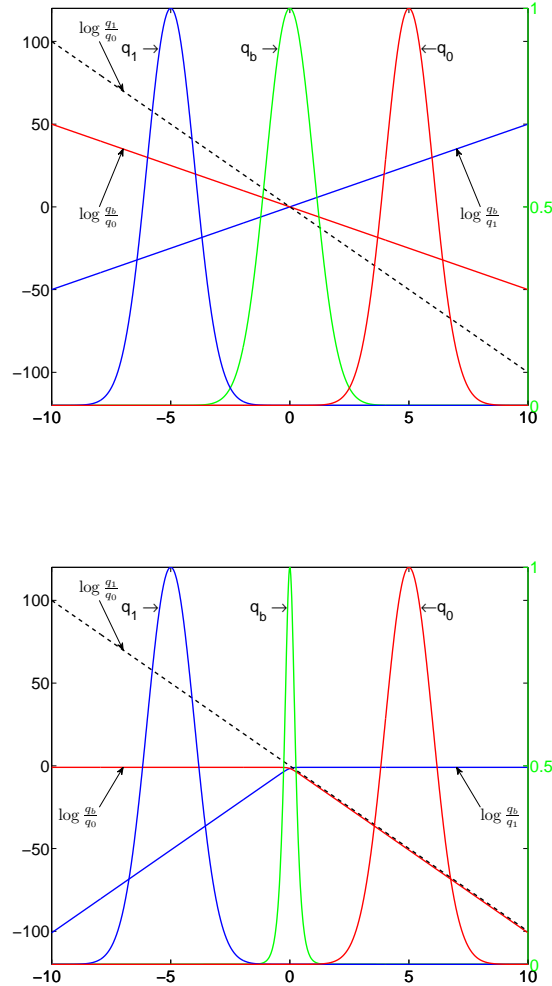


Figure 4.14: **Examples of bridge distribution.** Here q_1 and q_0 are unit variance normal distributions with mean -5 and 5 respectively. In the top plot, we display the geometric bridge distribution $q_b = \sqrt{q_0 q_1}$, which is simply an unnormalized standard normal distribution in this case (rescaled to have the same maximum as q_0 for clarity). In the bottom plot, we depict the harmonic mean bridge distribution $q_b = (\frac{1}{p_0} + \frac{1}{p_1})^{-1}$ for comparison. The other lines represent the value of the log of the weight function (units are given on the left axis) which will be used by the importance sampler vs. the bridge sampler. Notice that the highest weights for $\log \frac{q_1}{q_0}$ are twice as high as the ones for either $\log \frac{q_b}{q_0}$ or $\log \frac{q_b}{q_1}$ in the case of the geometric bridge. This suggests that bridge sampling with a geometric bridge should have lower variance than importance sampling. For the (asymptotically optimal) harmonic bridge, the weights are upper bounded by 1, and thus we expect its convergence properties to be even better.

approximation of q_1 as the proposal. Different choices of proposal distributions and bridges give well-known estimators for the marginal likelihood. For example, using $q_b = p_0$ (the known proposal distribution) yields $Z_1 = (\mathbb{E}_1 \left[\frac{p_0}{q_1} \right])^{-1}$, the *reciprocal sampling estimate* (RIS) of the marginal likelihood (Gelfand and Dey, 1994). Using the prior as the proposal $p_0(z) = p(z)$ in the RIS yields the *harmonic mean estimator* $p(x) = \mathbb{E}_1 \left[\frac{1}{p(z|x)} \right]^{-1}$ of Newton and Raftery (1994) mentioned earlier. Note that we mentioned earlier that a good property of the bridge q_b is to have thinner tails than p_0 and p_1 ; but in the case of the harmonic mean estimator, q_b is the prior, which normally has heavier tail than the posterior p_1 , hence explaining the poor behavior of this estimator.

Finally, one can extend the idea of bridge sampling to using multiple bridges: $\frac{Z_1}{Z_0} = \frac{Z_1}{Z_{b1}} \frac{Z_{b1}}{Z_{b2}} \dots \frac{Z_{bT}}{Z_0}$. Taking an infinite continuous limit yields *path sampling* (see Gelman and Meng, 1998). A seemingly related but fundamentally different approach is *annealed importance sampling* (AIS, Neal, 2001) which can be viewed as importance sampling on a higher-dimensional state space where auxiliary variables are introduced in order to make the proposal distribution closer to the target distribution.

4.B.1 Application to DiscLDA

We now describe how to apply the bridge sampling framework to estimate the posterior over labels $p(y|\mathbf{w}_d, \mathbb{T}, \Phi)$ for the DiscLDA model. As mentioned in Sec. 4.3.2, we estimate the ratio of posteriors for two different classes y_0 and y_1 by estimating the ratio Z_1/Z_0 where Z_c is the normalization constant for $q_c(\mathbf{z}_d) \doteq p(\mathbf{w}_d|\mathbf{z}_d, y_c, \mathbb{T}, \Phi)p(\mathbf{z}_d)$. Plugging the geometric bridge $q_b = \sqrt{q_0 q_1}$ in the bridge sampling estimate (4.32) with $M_0 = M_1 = M$ and using our notation of Sec. 4.A.1.2 yields:

$$\frac{Z_1}{Z_0} \approx \frac{\sum_{i=1}^M h_{10}(\mathbf{z}_d^{(i)0})}{\sum_{i=1}^M h_{01}(\mathbf{z}_d^{(i)1})} \quad (4.33)$$

where

$$h_{ab}(\mathbf{z}_d) \doteq \sqrt{\prod_n \frac{\Phi_{\mathbf{w}_{dn}z_{dn}}^{y_a}}{\Phi_{\mathbf{w}_{dn}z_{dn}}^{y_b}}}$$

and $\mathbf{z}_d^{(i)c}$ is sampled iid for $i = 1, \dots, M$ from $p(\mathbf{z}_d|y_c, \mathbf{w}_d, \mathbb{T}, \Phi)$. Note that in practice, we use Gibbs sampling to obtain $\mathbf{z}_d^{(i)c}$ and so the samples are actually correlated. We only use them to evaluate expectations, and so the correlation doesn't change the correctness of the result (it only reduces the effective sample size). As shown by [MacEachern and Berliner \(1994\)](#), there is actually no statistical gain to use a lag in the Gibbs samples to evaluate an expectation, and so we did use all the samples with no lag in our implementation of equation (4.33)¹⁴.

[Wallach et al. \(2009\)](#) mentions several alternative to estimate the marginal likelihood for the LDA model (which is similar in our case to $p(\mathbf{w}_d|\mathbf{z}_d, y_c, \mathbb{T}, \Phi)$). The marginal likelihood can be used to estimate our label posterior $p(y|\mathbf{w}_d, \mathbb{T}, \Phi)$ by estimating the marginal likelihood $p(\mathbf{w}_d|\mathbf{z}_d, y_c, \mathbb{T}, \Phi)$ for each y_c , including the prior $p(y_c)$ and renormalizing.

Using the harmonic mean (HM) estimator for the marginal likelihood yields an estimator for the label posterior of similar computational complexity as our bridge sampling (BS) estimate, but much worse accuracy in exploratory experiments that we did on synthetic documents of small length for which we can compute the true probability exactly. Moreover, the variance for HM was significantly higher than the one from the BS estimator on real documents. [Wallach et al. \(2009\)](#) mentioned that the AIS estimator for the marginal likelihood had one of the best accuracy. We tried their Matlab code for AIS in our exploratory experiments and obtained similar accuracy as the BS estimator but at a much higher computational cost. Namely,

¹⁴The only rationale to use a lag is when processing each sample is more computationally expensive than getting more samples, so one needs to focus on high quality samples. On the other hand, see [MacEachern and Peruggia \(2000\)](#) for an interesting counter-example where a *non-systematic* subsampling of the output of a Gibbs sampler could yield to a more efficient estimator.

with their default setting of 1,000 temperature steps, it takes about 20 hours to do inference on the test set of the “1 vs. 20” binary classification task described in Table 4.2 (80k tokens), using the model of Sec. 4.4.3 ($K = 60$ topics). In contrast, our C implementation for bridge sampling with $M = 2.5k$ takes about 2 minutes on this dataset.

Chapter 5

Conclusions

5.1 Summary

In this thesis, we have extended the range of applicability of discriminative learning methods to exploit different types of structure. In Chapter 2, we have developed a scalable max-margin learning algorithm which could be applied to a wide range of structured output problems, including some with a combinatorial structure, for which a maximum-likelihood approach would be intractable in contrast. We applied this framework in Chapter 3 to the word alignment problem and showed how the discriminative approach enabled us to use a rich set of features, including the predictions of other models. In particular, we could leverage features such as fertility and first-order interactions which to date have been exploited only in generative approaches. With those improvements, we obtained a word aligner with state-of-the-art performance. In Chapter 4, we proposed an approach to exploit the latent variable structure of documents in the context of text classification. We presented DiscLDA, a discriminative variant of LDA in which a transformation matrix on the LDA parameters is learned via maximization of the conditional likelihood. Our experiments showed that

we could obtain a useful novel reduced representation of documents which maintained predictive power, while having an interesting interpretability through the learned discriminative topics.

5.2 Future Directions

Following on the themes developed in this thesis, we identify the following areas of future work:

- *Theoretical understanding of generative vs. discriminative learning.* One motivation for this thesis was that discriminative learning seems to perform better in practice for pure supervised classification. On the other hand, whether this is the case in general, and what would be the underlying theoretical principles are still not clearly understood. On the topic of supervised topic models, there have now been several proposals for discriminative versions of topic models as well as generative versions. A more systematic comparison could shed some light on this question.
- *Machine translation.* We obtained impressive results in terms of word alignment quality, but an important question for natural language processing is to include these in a machine translation pipeline and see whether the translation accuracy improves as well.
- *Semi-supervised learning.* DiscLDA being embedded in a graphical model, it could be interesting to apply it in the semi-supervised learning setting to leverage the information from a large unlabeled collection of documents in addition to our small set of labeled documents.
- *Variational approach to DiscLDA.* We have used a sampling approach for Dis-

cLDA given its simplicity and its wide use in the probabilistic topic modeling literature. On the other hand, the level of noise in our results made it apparent that a variational approach would probably be more suitable for discriminative learning, by exchanging the sampling noise with a (hopefully) small deterministic bias of the method.

- *Supervised Topic Models.* There are now several proposals for supervised topic models: some are trained generatively, such as in the supervised LDA model of [Blei and McAuliffe \(2008\)](#) or in the Dirichlet-multinomial model of [Mimno and McCallum \(2008\)](#); others are trained discriminatively, such as the MedLDA model of [Zhu et al. \(2009\)](#) or our DiscLDA model. A systematic experimental comparison would be useful to obtain some insight into the relative advantages and disadvantages of these various approaches.

Bibliography

- Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov support vector machines. In *International Conference on Machine Learning*. ACM Press, New York, 2003. [5](#), [7](#)
- D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Y. Ng. Discriminative learning of Markov random fields for segmentation of 3D scan data. In *International Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Washington, DC, 2005. [6](#), [10](#), [35](#)
- A. Asuncion, M. Welling, P. Smyth, and Y. Teh. On smoothing and inference for topic models. In *Uncertainty in Artificial Intelligence*, 2009. [100](#)
- P. Baldi, J. Cheng, and A. Vullo. Large-scale prediction of disulphide bond connectivity. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2005. [4](#)
- P. L. Bartlett, M. Collins, B. Taskar, and D. McAllester. Exponentiated gradient algorithms for large-margin structured classification. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2005. [7](#), [23](#)
- S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:509–522, 2002. [4](#)
- T. L. Berg, A. C. Berg, J. Edwards, M. Maire, R. White, Y. W. Teh, E. Learned-Miller, and D. A. Forsyth. Names and faces in the news. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Washington, DC, 2004. [69](#)
- A. Bernal, K. Crammer, A. Hatzigeorgiou, and F. Pereira. Global discriminative learning for higher-accuracy computational gene prediction. *PLoS Comput Biol*, 3(3):e54, 03 2007. [44](#)

BIBLIOGRAPHY

- D. P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, Belmont, MA, 1998. [25](#)
- D. P. Bertsekas, L. C. Polymenakos, and P. Tseng. An ϵ -relaxation method for separable convex cost network flow problems. *SIAM Journal on Optimization*, 7(3):853–870, 1997. [25](#)
- D. Blei and J. McAuliffe. Supervised topic models. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008. [97](#), [98](#), [119](#)
- D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. [69](#), [71](#), [72](#), [77](#), [78](#)
- L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. MIT Press, Cambridge, MA, 2008. [44](#)
- S. Boyd and A. Mutapcic. Subgradient methods. Downloaded from http://www.stanford.edu/class/ee364b/notes/subgrad_method_notes.pdf, January 2007. Notes for EE364b, Stanford University. [45](#)
- S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. [23](#), [24](#)
- P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990. [50](#), [57](#)
- B. Carlson. *Special Functions of Applied Mathematics*. Academic Press, New York, New York, 1977. [107](#)
- C. Chekuri, S. Khanna, J. Naor, and L. Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, Washington, DC, 2001. [9](#), [10](#)
- C. Chekuri, S. Khanna, J. Naor, and L. Zosin. A linear programming formulation and approximation algorithms for the metric labeling problem. *SIAM Journal on Discrete Mathematics*, 18(3):606–635, 2005. [9](#), [10](#)
- F. Chiaromonte and R. D. Cook. Sufficient dimension reduction and graphics in regression. *Annals of the Institute of Statistical Mathematics*, 54(4):768–795, 2002. [69](#)

BIBLIOGRAPHY

- M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Philadelphia, PA, 2002. [5](#), [14](#), [34](#), [36](#), [51](#)
- K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003. [44](#)
- K. Crammer, R. McDonald, and F. Pereira. Scalable large-margin online learning for structured classification. Technical report, Department of Computer and Information Science, University of Pennsylvania, 2005. [44](#)
- K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006. [44](#)
- H. Daumé III, J. Langford, and D. Marcu. Search-based structured prediction. *Machine Learning*, 75(3), June 2009. [5](#)
- R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*. Cambridge University Press, UK, 1998. [14](#)
- L. Fei-fei and P. Perona. A Bayesian hierarchical model for learning natural scene categories. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, CA, 2005. [98](#)
- T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 304–311. ACM, New York, NY, USA, 2008. [43](#)
- P. Flaherty, G. Giaever, J. Kumm, M. I. Jordan, and A. P. Arkin. A latent variable model for chemogenomic profiling. *Bioinformatics*, 21:3286–3293, 2005. [98](#)
- K. Fukumizu, F. R. Bach, and M. I. Jordan. Kernel dimension reduction in regression. *Annals of Statistics*, 37:1871–1905, 2009. [69](#)
- A. E. Gelfand and D. K. Dey. Bayesian model choice: Asymptotics and exact calculations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 56(3): 501–514, 1994. [114](#)
- A. Gelman and X.-L. Meng. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical Science*, 13(2):163–185, 1998. [114](#)

BIBLIOGRAPHY

- D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society B*, 51:271–279, 1989. [10](#)
- T. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235, 2004. [69](#), [72](#), [77](#), [78](#), [79](#), [80](#), [84](#)
- F. Guerriero and P. Tseng. Implementation and test of auction methods for solving generalized network flow problems with separable convex cost. *Journal of Optimization Theory and Applications*, 115:113–144, 2002. [25](#), [36](#)
- T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004. [35](#)
- E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007. [45](#)
- B. He and L.-Z. Liao. Improvements of some projection methods for monotone non-linear variational inequalities. *Journal of Optimization Theory and Applications*, 112:111–128, 2002. [20](#), [22](#)
- T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press, Cambridge, MA, 1999. [98](#)
- M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22:1087–1116, 1993. [6](#), [14](#)
- M. I. Jordan, editor. *Learning in Graphical Models*. MIT Press, Cambridge, MA, USA, 1999. [2](#)
- P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *Human Language Technology–North American Association for Computational Linguistics*, 2003. [67](#)
- V. Kolmogorov and M. Wainwright. On the optimality of tree-reweighted max-product message passing. In *Uncertainty in Artificial Intelligence: Proceedings of the Twenty-First Conference*. Morgan Kaufmann, San Mateo, CA, 2005. [10](#)
- V. Kolmogorov and R. Zabih. What energy functions can be minimized using graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:147–159, 2004. [10](#)

BIBLIOGRAPHY

- G. M. Korpelevich. The extragradient method for finding saddle points and other problems. *Ekonomika i Matematicheskie Metody*, 12:747–756, 1976. [20](#)
- S. Kumar and M. Hebert. Discriminative fields for modeling spatial dependencies in natural images. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2004. [6](#)
- S. Lacoste-Julien, B. Taskar, D. Klein, and M. I. Jordan. Word alignment via quadratic assignment. In *Human Language Technology–North American Association for Computational Linguistics*, New York, NY, 2006. [3](#)
- S. Lacoste-Julien, F. Sha, and M. I. Jordan. DiscLDA: Discriminative learning for dimensionality reduction and classification. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 897–904, 2009. [3](#), [80](#)
- J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*. Morgan Kaufmann, San Mateo, CA, 2001. [2](#), [5](#)
- J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: Representation and clique selection. In *International Conference on Machine Learning*. ACM Press, New York, 2004. [5](#)
- H. Larochelle and Y. Bengio. Classification using discriminative restricted Boltzmann machines. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 536–543, New York, NY, USA, 2008. ACM. [98](#)
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998. [6](#)
- Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. In G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar, editors, *Predicting Structured Data*. MIT Press, 2006. [5](#)
- P. Liang and M. I. Jordan. An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In A. McCallum and S. Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 584–591. Omnipress, 2008. [2](#)
- P. Liang, B. Taskar, and D. Klein. Alignment by agreement. In *Human Language Technology–North American Association for Computational Linguistics*, 2006. [53](#), [65](#)

BIBLIOGRAPHY

- J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, New York, NY, USA, 2001. [84](#), [111](#)
- S. N. MacEachern and M. Peruggia. Subsampling the gibbs sampler: variance reduction. *Statistics & Probability Letters*, 47(1):91–98, 2000. [115](#)
- S. N. MacEacherna and L. M. Berliner. Subsampling the Gibbs sampler. *The American Statistician*, 48(3):188–190, 1994. [115](#)
- C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999. [14](#)
- E. Matusov, R. Zens, and H. Ney. Symmetric word alignments for statistical machine translation. In *Proceedings of the Twentieth International Conference on Computational Linguistics*, Geneva, Switzerland, 2004. [4](#)
- R. McDonald, K. Crammer, and F. Pereira. Online large-margin training of dependency parsers. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98. ACL, Morristown, NJ, USA, 2005. [44](#)
- X.-L. Meng and W. H. Wong. Simulating ratios of normalizing constants via a simple identity: a theoretical exploration. *Statistica Sinica*, 6:831–860, 1996. [80](#), [109](#), [112](#)
- R. Mihalcea and T. Pedersen. An evaluation exercise for word alignment. In *Human Language Technology–North American Association for Computational Linguistics*, Edmonton, Canada, 2003. [38](#)
- D. Mimno and A. McCallum. Topic models conditioned on arbitrary features with Dirichlet-multinomial regression. In *Proceedings of the 24th Annual Conference on Uncertainty in Artificial Intelligence*, Helsinki, Finland, 2008. [98](#), [119](#)
- R. C. Moore. A discriminative framework for bilingual word alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2005. [51](#), [63](#)
- R. M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001. [114](#)
- Y. Nesterov. Dual extrapolation and its application for solving variational inequalities and related problems. Technical report, CORE, Catholic University of Louvain, 2003. [2](#), [7](#), [20](#), [21](#), [22](#), [23](#), [26](#), [46](#)
- M. A. Newton and A. E. Raftery. Approximate bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society. Series B (Methodological)*, 56(1):3–48, 1994. [111](#), [114](#)

BIBLIOGRAPHY

- A. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press. [2](#)
- F. Och and H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52, 2003. [38](#), [39](#), [50](#), [57](#), [63](#)
- J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Sch editor, *Advances in Kernel Methods—Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, 1999. [6](#)
- N. Ratliff, J. A. Bagnell, and M. Zinkevich. (online) subgradient methods for structured prediction. In *Eleventh International Conference on Artificial Intelligence and Statistics (AISTats)*, March 2007. [44](#), [45](#)
- M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence*, Banff, Canada, 2004. [97](#)
- S. Rosset. Tracking curved regularized optimization solution paths. In *Advances in Neural Information Processing Systems*. [35](#)
- A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, Berlin, 2003. [11](#), [13](#), [54](#)
- S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated sub-Gradient SOLver for SVM. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 807–814. ACM, New York, NY, USA, 2007. [45](#)
- B. Taskar. *Learning Structured Prediction Models: A Large Margin Approach*. PhD thesis, Stanford University, 2004. [6](#), [52](#), [54](#)
- B. Taskar, V. Chatalbashev, and D. Koller. Learning associative Markov networks. In *International Conference on Machine Learning*. ACM Press, New York, 2004a. [6](#), [10](#), [18](#)
- B. Taskar, C. Guestrin, and D. Koller. Max margin Markov networks. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2004b. [2](#), [5](#), [6](#), [14](#), [15](#), [18](#), [44](#)

BIBLIOGRAPHY

- B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: a large margin approach. In *International Conference on Machine Learning*. ACM Press, New York, 2005a. [6](#), [18](#)
- B. Taskar, S. Lacoste-Julien, and D. Klein. A discriminative matching approach to word alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Vancouver, CA, 2005b. [39](#), [51](#), [64](#)
- B. Taskar, S. Lacoste-Julien, and M. I. Jordan. Structured prediction via the extragradient method. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2006a. [20](#)
- B. Taskar, S. Lacoste-Julien, and M. I. Jordan. Structured prediction, dual extragradient and Bregman projections. *Journal of Machine Learning Research*, 7:1627–1653, 2006b. [2](#)
- Y. W. Teh, D. Newman, and M. Welling. A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems*, volume 19, 2007. [78](#)
- I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005. [7](#), [14](#), [43](#)
- L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979. [14](#)
- L. J. P. van der Maaten and G. E. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. [85](#)
- S. Vogel, H. Ney, and C. Tillmann. HMM-based word alignment in statistical translation. In *International Conference on Computational Linguistics*, pages 836–841, 1996. [57](#)
- M. Wainwright, T. Jaakkola, and A. Willsky. Map estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. In *Proceedings of the Allerton Conference on Communication, Control and Computing*, Allerton, IL, 2002. [9](#)
- H. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno. Evaluation methods for topic models. In L. Bottou and M. Littman, editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 1105–1112, Montreal, June 2009. Omnipress. [115](#)

BIBLIOGRAPHY

- J. Zhu, A. Ahmed, and E. Xing. MedLDA: Maximum margin supervised topic models for regression and classification. In L. Bottou and M. Littman, editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 1257–1264. Omnipress, 2009. [98](#), [119](#)

Index

- Bregman divergence, 27
- bridge distribution, 111
- bridge sampling, 80, 109
- dimensionality reduction, 69, 81
- DiscLDA model, 72
 - learning algorithm, 78
- divergence function
 - Euclidean, 23
 - non-Euclidean, *see* Bregman divergence
- dual extragradient
 - convergence, 22, 43
 - Euclidean version, 21
 - kernels, 32
 - memory-efficient version, 33
 - non-Euclidean version, 26
- gap function, 22
- gradient matching, 80, 103
- gradient-based algorithms, 6, 44
- hinge loss, 15
- LDA model, 71
- Linear Program (LP)
 - for figure-ground segmentation, 11
 - for general prediction problems, 17
 - for loss-augmented inference, 17
 - dual of, 18
 - for matchings, 13
 - loss-augmented inference, 16
 - for tree-structured Markov networks, 9
 - loss-augmented inference problem, 16
 - as a LP, 17
 - dual LP, 18
 - for matchings, 16
 - maximum conditional likelihood, 5, 14, 76
 - maximum-margin learning, 5, 15
 - polynomial QP formulation, 19
 - saddle-point formulation, *see* saddle-point problem
 - min-cost quadratic flow, 25
 - projection
 - Euclidean, 20
 - for matchings, 25
 - non-Euclidean, *see* proximal step operator
 - on the simplex, 29, 30
 - proximal step operator, 26
 - for L_1 regularization, 29
 - for tree-structured marginals, 30
 - regularization path, 35, 43
 - restricted gap function, 23
 - saddle-point problem, 17, 19, 61
 - structured output models, 5, 8
 - figure-ground segmentation, 10, 35
 - Markov networks with submodular potentials, 10

INDEX

- matchings, [12](#), [53](#)
- tree-structured Markov networks, [8](#)
- word alignment, [38](#), [53](#)
- submodular potentials, [10](#)