Fall 12-22-2010

Posterior Regularization for Learning with Side Information and Weak Supervision

Kuzman Ganchev
University of Pennsylvania, kuzman@seas.upenn.edu

Posterior Regularization for Learning with Side Information and Weak Supervision

Abstract
Supervised machine learning techniques have been very successful for a variety of tasks and domains including natural language processing, computer vision, and computational biology. Unfortunately, their use often requires creation of large problem-specific training corpora that can make these methods prohibitively expensive. At the same time, we often have access to external problem-specific information that we cannot alway easily incorporate. We might know how to solve the problem in another domain (e.g. for a different language); we might have access to cheap but noisy training data; or a domain expert might be available who would be able to guide a human learner much more efficiently than by simply creating an IID training corpus. A key challenge for weakly supervised learning is then how to incorporate such kinds of auxiliary information arising from indirect supervision.

In this thesis, we present Posterior Regularization, a probabilistic framework for structured, weakly supervised learning. Posterior Regularization is applicable to probabilistic models with latent variables and exports a language for specifying constraints or preferences about posterior distributions of latent variables. We show that this language is powerful enough to specify realistic prior knowledge for a variety applications in natural language processing. Additionally, because Posterior Regularization separates model complexity from the complexity of structural constraints, it can be used for structured problems with relatively little computational overhead. We apply Posterior Regularization to several problems in natural language processing including word alignment for machine translation, transfer of linguistic resources across languages and grammar induction. Additionally, we find that we can apply Posterior Regularization to the problem of multi-view learning, achieving particularly good results for transfer learning. We also explore the theoretical relationship between Posterior Regularization and other proposed frameworks for encoding this kind of prior knowledge, and show a close relationship to Constraint Driven Learning as well as to Generalized Expectation Constraints.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Computer and Information Science

First Advisor
Fernando Pereira

Second Advisor
Ben Taskar

Keywords
Posterior Regularization Framework, Unsupervised Learning, Latent Variable Models, Prior Knowledge, Natural Language Processing, Machine Learning, Partial Supervision

Subject Categories
Artificial Intelligence and Robotics | Computer Sciences | Statistical Models

POSTERIOR REGULARIZATION FOR LEARNING WITH SIDE INFORMATION AND WEAK SUPERVISION

Kuzman Ganchev

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2010

Fernando Pereira, Professor of Computer and Information Science
Supervisor of Dissertation

Ben Taskar, Assistant Professor of Computer and Information Science
Supervisor of Dissertation

Jianbo Shi, Associate Professor of Computer and Information Science
Graduate Group Chairperson

Dissertation Committee

Michael Collins, Associate Professor of Computer Science, MIT

Mark Liberman, Professor of Phonetics Department of Linguistics

Mitch Marcus, Professor of Computer and Information Science

Lyle Ungar, Associate Professor of Computer and Information Science

Posterior Regularization for Learning with Side Information and Weak Supervision

This thesis is dedicated to my parents, who have given me everything.

Anything I achieve, you have achieved.

Acknowledgements

Thanks . . .

To my advisors Fernando Pereira and Ben Taskar. Fernando was my first advisor at Penn and has been a source of inspiration from the start. It seems that no matter how well I think I understand a problem, Fernando can shed new light on it. His advice is always invaluable, and more often than not surprising. He seems to know relevant literature in areas I would never have believed to be related, and to find bugs, as if by magic, in code he has never seen. Ben was my advisor since shortly after he joined Penn. None of the work presented here would have been conceived, implemented, tested, or described were it not for him. "Ask Ben" seems to be a universal solution, and one of the hardest things about leaving will be losing the ability to drop by his office to find out how to formalize, optimize and visualize my ideas. Both Ben and Fernando are amazing people that have created a friendly, warm, stress-free atmosphere that has made my years at Penn so enjoyable, and I might never be able to repay them fully for all they have done for me.

To my committee: Mitch Marcus, Mike Collins, Mark Liberman, and Lyle Ungar for their feedback, and support. Thanks to Mitch for agreeing to chair my committee and for telling me what isn't obvious – Chapter 5 was largely his idea. Mitch was never formally my advisor, but he often played the part and I always felt welcome to ask for his advice and support. Thanks to Mike for the very helpful, detailed comments and suggestions on both the proposal and the thesis documents, and for agreeing to come in person both times. Mark provided much needed linguistic perspective and suggested new applications and avenues, that I would never have thought of. Thanks to Lyle for the very helpful comments both at the proposal and at the defense. Without your direction this dissertation would have been much less than it has become.

To Michael Kearns for showing me the world of finance, what it means to be clear, and that it is possible to finish before the deadline. To Kiril Simov for giving me the opportunity to do research in my native Bulgaria, where the work on Chapter 10 took place. To Tia Newhall, who introduced me to research when I was an undergraduate, and who is my first co-author. And to Richard Wicentowski, who introduced me to natural language processing.

To the co-authors of the work in this dissertation. To João Graça, who is a partner for the work in almost every chapter of this thesis, and who is usually responsible for more than half of the energy in every project. To Jennifer Gillenwater who is a partner on all the parsing chapters, and who writes the neatest, prettiest code I have ever seen. To John Blitzer, who is a partner on the multi-view chapter (John – gl, hf). To my other collaborators, and especially to Kedar Bellare, Steven Carroll, Koby Crammer, Mark Dredze, Ryan Gabbard, Georgi Georgiev, Yang Jin, Alex Kulesza, Qian Liu, Mark Mandel, Gideon Mann, Andrew McCallum, Ryan McDonald, Vassil Momchev, Preslav Nakov, Yuriy Nevmyvaka, Deyan Peychev, Angus Roberts, Partha Pratim Talukdar, Jinsong Tan, Jennifer Wortman Vaughan, and Peter White.

To the administrative staff, who are truly remarkable and especially to Mike Felker, who has been voted "most useful person" in every informal poll. Thank you for running the place so smoothly, keeping our lives simple, and always finding a way to resolve our crises.

To Aaron, Alex, Alex, Alex, Axel, Brian, Cheryl, David, Dimo, Drew, Emily, Gabe, Galia, Hannah, Iliana, Irena, Jeff, Jenn, Jenny, João, Julie, Karl, Lauren, Mike, Nick, Partha, Qian, Rachel, Sophie, Wynn, for all the love, laughter, hugs, parties, dinners, drinks, mafia games, D&D&D, zip-boing. Thank you for making these past six years so great.

Finally, to my mother, who nurtured my creativity and always cleaned up after my early "science experiments"; to my father who taught me that everything is possible and to strive for the best; to my brother without whose help I would never have gotten into any university and who was always kind even when I was a brat; and to my love who makes struggles bearable and triumphs sweeter every single day.

ABSTRACT

Posterior Regularization for Learning with Side Information and Weak Supervision

Kuzman Ganchev

Supervisors: Fernando Pereira and Ben Taskar

Supervised machine learning techniques have been very successful for a variety of tasks and domains including natural language processing, computer vision, and computational biology. Unfortunately, their use often requires creation of large problem-specific training corpora that can make these methods prohibitively expensive. At the same time, we often have access to external problem-specific information that we cannot alway easily incorporate. We might know how to solve the problem in another domain (e.g. for a different language); we might have access to cheap but noisy training data; or a domain expert might be available who would be able to guide a human learner much more efficiently than by simply creating an IID training corpus. A key challenge for weakly supervised learning is then how to incorporate such kinds of auxiliary information arising from indirect supervision.

In this thesis, we present Posterior Regularization, a probabilistic framework for structured, weakly supervised learning. Posterior Regularization is applicable to probabilistic models with latent variables and exports a language for specifying constraints or preferences about posterior distributions of latent variables. We show that this language is powerful enough to specify realistic prior knowledge for a variety applications in natural language processing. Additionally, because Posterior Regularization *separates* model complexity from the complexity of structural constraints, it can be used for structured problems with relatively little computational overhead. We apply Posterior Regularization to several problems in natural language processing including word alignment for machine translation, transfer of linguistic resources across languages and grammar induction. Additionally, we find that we can apply Posterior Regularization to the problem of multi-view learning, achieving particularly good results for transfer learning. We also explore the theoretical re-

lationship between Posterior Regularization and other proposed frameworks for encoding this kind of prior knowledge, and show a close relationship to Constraint Driven Learning as well as to Generalized Expectation Constraints.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning in general and statistical models in particular have found applications in a wide variety of domains, across a range of disciplines including natural language processing, computer vision, signal processing, computational finance medical image analysis and computational biology to name just a few. For many of these application domains the most successful models use supervised machine learning approaches that require large quantities of annotated data in order to build models able to perform well. [1]

In some situations, the annotations necessary for applying supervised learning algorithms occur naturally or can be obtained relatively inexpensively. For example, if we would like to predict the price and volatility of stocks, bonds or futures contracts, we can obtain historical data relatively inexpensively. This data can then be used to train predictors for use in the future. Similarly, machine translation systems are typically trained using already available corpora such as parliamentary proceedings that have to be available in multiple languages by law. In contrast to these relatively data-abundant applications, for the vast majority of applications the successful application of machine learning techniques requires expensive manual annotation. For example, in order to train a state of the art named entity recognition system thousands of sentences must be annotated for the entities of interest. A similar situation exists for other machine learning applications such as syntactic analysis of natural language, coreference resolution, relation extraction, object recognition, gene finding, handwriting recognition and speech recognition.

[1] Appendix A provides some background on probabilistic models.

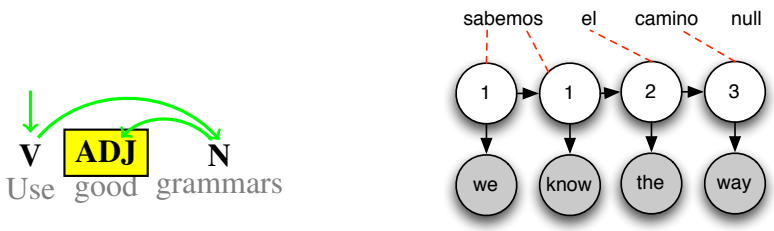


Figure 1.1: Two examples of structured models we will use in this work. Left: a dependency tree representation of syntax. The hidden variables are the identities of the edges that compose the tree. Right: a hidden Markov model used for word alignment. The hidden variables are the identities of the source language words that are translations of the target language words.

This annotation process is often the most time-consuming and expensive part of the construction of usable model. For example, 400 hours were needed to label a single hour of speech at the phonetic level [Greenberg, 1996]. The Penn Chinese Treebank project released the first version of its 4,000 sentences two years after the project began [Hwa et al., 2005]. Furthermore, to achieve optimal performance, we need a separate corpus for each domain of interest. For example, syntactic parsers trained on news perform poorly on biomedical test [Dredze et al., 2007]. As the number of tasks, languages and target domains of interest increases hand-labeling quickly becomes prohibitively slow and expensive.

For many problems where it is expensive to create annotations, unannotated data are abundant. For example, natural language text in many domains is widely available, sequenced genomes for many organisms can be downloaded from public databases, and large collections of images are much easier to obtain than annotated images. In order to exploit this inexpensive data, a variety of unsupervised machine learning approaches have been devised. In unsupervised problems where data has sequential, recursive, spatial, relational, and other kinds of structure, we often employ structured statistical models with latent variables to tease apart the underlying dependencies and induce meaningful semantic categories. Unsupervised part-of-speech and grammar induction, and word and phrase alignment for statistical machine translation in natural language processing are examples of such aims. Generative models (probabilistic grammars, graphical models, etc.) are usually estimated by maximizing the likelihood of the observed data by marginalizing over the

hidden variables, typically via the Expectation Maximization (EM) algorithm. Figure 1.1 shows a couple of examples of the model structures that we deal with in this dissertation.

Because of computational and statistical concerns, generative models used in practice are very simplistic models of the underlying phenomena; for example, the syntactic structure of language or the language translation process. A pernicious problem with such models is that marginal likelihood may not guide the model towards the intended role for the latent variables, instead focusing on explaining irrelevant but common correlations in the data. Since we are mostly interested in the distribution of the latent variables in the hope that they capture *intended* regularities without direct supervision, controlling this latent distribution is critical. Less direct methods such as clever initialization, ad hoc procedural modifications, and complex data transformations are often used to affect the posteriors of latent variables in a desired manner. As an example, in the problem of part of speech induction, the goal is to derive a set of syntactic categories from unannotated text. Because of computational complexity and our limited understanding of how children learn syntactic categories, this problem is typically solved using maximum likelihood training and a hidden Markov model. As we see in Figure 1.2 (left panel), maximum likelihood training results in very high ambiguity for each word. Depending on context, the hidden Markov model might label the word "China" as a "noun", "verb", or "preposition."

A key challenge for structured, weakly supervised learning is developing a flexible, declarative framework for expressing structural constraints on latent variables arising from prior knowledge and indirect supervision. Structured models have the ability to capture a very rich array of possible relationships, but adding complexity to the model often leads to intractable inference. In this dissertation, we present the posterior regularization (PR) framework, which *separates* model complexity from the complexity of structural constraints it is desired to satisfy. Unlike parametric regularization in a Bayesian framework, our approach incorporates data-dependent constraints that are easy to encode as information about model *posteriors* on the observed data, but may be difficult to encode as information about model parameters through Bayesian *priors*. In the right panel of Figure 1.2 we see that by imposing an appropriate penalty for the ambiguity of each word, we effectively mitigate the high-ambiguity problem, and produce a cleaner set of grammatical

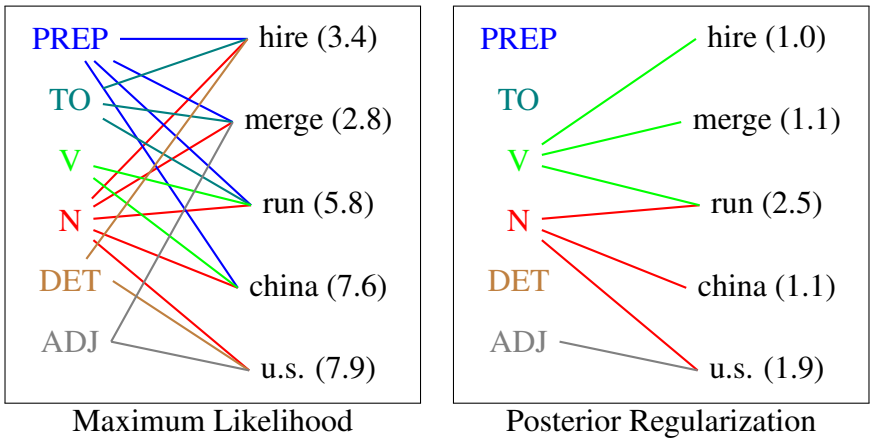


Figure 1.2: Example part of speech induction application. In each panel words are on the right and parts of speech on the left. A link mean that the word was automatically tagged with that part of speech in some context. The numbers in parentheses indicate a measure of the tag-ambiguity of each word (see Chapter 9 for details). Left: conventional maximum likelihood training. Right: the method proposed in this work. See text for explanation.

categories. The experiments that produced Figure 1.2 are described in Chapter 9. Chapters 5 and 6-10 describe a variety of such useful prior knowledge constraints in several application domains.

1.1 Contributions

This thesis deals with the problem of incorporating prior knowledge into unsupervised and semi-supervised learning for structured and unstructured models. We focus on models where exact efficient inference is possible, and describe a framework for efficiently incorporating prior knowledge into both generative and discriminative models, both for structured and unstructured data. Specifically, we present:

- A flexible, declarative framework for structured, weakly supervised learning via posterior regularization.
- An efficient algorithm for model estimation with posterior regularization.
- An extensive evaluation of different types of constraints in several domains: multi-view learning, cross-lingual dependency grammar induction, unsupervised part-of-

speech induction, unsupervised grammar induction and bitext word alignment.

- A detailed explanation of the connections between several other recent proposals for weak supervision, including structured constraint-driven learning [Chang et al., 2007], generalized expectation criteria [Mann and McCallum, 2008, 2007] and Bayesian measurements [Liang et al., 2009].

1.2 Thesis Overview

This section describes the general organization of the thesis. Part I of the document describes the mathematical formulation of the posterior regularization framework, tries to give some intuitions about the kinds of knowledge that can efficiently be incorporated into the framework for a variety of models, and describes how posterior regularization is related to a few recently proposed frameworks that try to encode similar kinds of prior knowledge. Part II of the document describes applications of posterior regularization for with associated experimental results. Chapter 11 concludes the dissertation.

1.2.1 Mathematical Formulation, Intuitions and Related Methods

Chapter 2 describes the posterior regularization (PR) framework as a mathematical formulation. Section 2.2 describes the objective we optimize and Sections 2.5 and 2.7 describe a simple algorithm to perform the optimization for generative and discriminative models. Chapter 3 previews the applications described in Part II. Chapter 4 relates the PR framework to several related methods: Constraint Driven Learning [Chang et al., 2007, 2008] in Section 4.1, Generalized Expectation Criteria [Mann and McCallum, 2007, 2008, 2010] in Section 4.2 and Bayesian Measurements [Liang et al., 2009] in Section 4.3. Figure 4.2 summarizes the relationship between these frameworks. Chapter 5 attempts to give an intuition for the kinds of prior knowledge that can be efficiently encoded in the PR framework, as well as how to encode them.

1.2.2 Applications and Experimental Results

In Part II of the thesis, we present a series of experiments using the PR framework in a variety of natural language application domains. Chapter 6 focuses on the problem of unsupervised statistical word alignment, and trying to enforce that alignments should usually be bijective and symmetric. Chapter 7 describes how the PR framework can be used for multi-view learning, resulting in a closed-form projection and a Bhattacharyya distance co-regularizer in two view learning. Chapter 8 describes experiments in projecting a dependency grammar from one language to another by using a bilingual text. Chapters 9 and 10 describe how PR can be used to induce a sparsity structure for the induction of syntactic analyses: part-of-speech induction in Chapter 9 and dependency grammar induction in Chapter 10.

Part I

Mathematical Formulation, Intuitions, Related Methods

Chapter 2

Posterior Regularization Framework

In this chapter we describe the posterior regularization framework, which incorporates side-information into parameter estimation in the form of linear constraints on posterior expectations. [1] As we will show, this allows tractable learning and inference even when the constraints would be intractable to encode directly in the model parameters. By defining a flexible language for specifying diverse types of problem-specific prior knowledge, we make the framework applicable to a wide variety of probabilistic models, both generative and discriminative. In Sections 2.1-2.6 we will focus on generative models, and describe the case of discriminative models in Section 2.7. We will use a problem from natural language processing as a running example in the exposition:

> **Running Example** *The task is part-of-speech (POS) tagging with limited or no training data. Suppose we know that each sentence should have at least one verb and at least one noun, and would like our model to capture this constraint on the unlabeled sentences. The model we will be using is a first-order hidden Markov model (HMM).*

We describe four other applications with empirical results in Chapters 6-10, but it will be easier to illustrate key concepts using this simple example.

[1] This chapter is based on Graça et al. [2007], Ganchev et al. [2010].

2.1 Preliminaries and Notation

We assume that there is a natural division of variables into "input" variables $\mathbf{x}$ and "target" variables $\mathbf{y}$ for each data instance, where $\mathbf{x}$'s are always observed. We denote the set of all instances of unlabeled data as $\mathbf{X}$. In case of semi-supervised learning, we have some labeled data as well, and we will use the notation $(\mathbf{X}_L, \mathbf{Y}_L)$ to denote all the labeled instances.

The starting point for using the PR framework is a probabilistic model. Let θ be the parameters of the model. For now we assume a generative model $p_\theta(\mathbf{x}, \mathbf{y})$, and we use $\mathcal{L}(\theta) = \log p_\theta(\mathbf{X}_L, \mathbf{Y}_L) + \log \sum_{\mathbf{Y}} p_\theta(\mathbf{X}, \mathbf{Y}) + \log p(\theta)$ to denote the parameter-regularized log-likelihood of the data.

Running Example *In the POS tagging example from above, we would use* $\mathbf{x} = \{x_1, x_2, \ldots x_{|\mathbf{x}|}\}$ *to denote a sentence (i.e. a sequence of words) and* $\mathbf{y} = \{y_1, y_2, \ldots y_{|\mathbf{x}|}\}$ *to denote a possible POS assignment. Using an HMM, it is defined in the normal way as:*

$$p_\theta(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{|\mathbf{x}|} \; p_\theta(y_i|y_{i-1}) \;\; p_\theta(x_i|y_i)$$

with θ *representing the multinomial distributions directly, and where* $p_\theta(y_1|y_0) = p_\theta(y_1)$ *represents a set of initial probabilities. Suppose we have a small labeled corpus and a larger unlabeled corpus. For a generative model such as an HMM, the log-likelihood (+ log-prior) is:*

$$\mathcal{L}(\theta) = \log p_\theta(\mathbf{X}_L, \mathbf{Y}_L) + \log \sum_{\mathbf{Y}} p_\theta(\mathbf{X}, \mathbf{Y}) + \log p(\theta),$$

where corpus probabilities are products over instances: $p_\theta(\mathbf{X}_L, \mathbf{Y}_L) = \prod p_\theta(\mathbf{x}, \mathbf{y})$ *and analogously for* $\mathbf{X}_L, \mathbf{Y}_L$*; and where* $p(\theta)$ *is a prior distribution over the parameters* θ*.*

Symbol	Meaning
$\mathbf{x}$	(observed) input variables for a particular example
$\mathbf{y}$	(usually hidden) output variables for a particular example
$\mathbf{X}, \mathbf{Y}$	$\mathbf{x}$ and $\mathbf{y}$ for the entire unlabeled portion of the corpus
$\mathbf{X}_L, \mathbf{Y}_L$	$\mathbf{x}$ and $\mathbf{y}$ for the entire labeled portion of the corpus (possibly empty)
$p_\theta(\mathbf{x}, \mathbf{y})$	a generative, joint model with parameters θ
$\mathcal{L}(\theta)$	data log-likelihood and parameter prior: $\log p_\theta(\mathbf{X}_L, \mathbf{Y}_L) + \log \sum_{\mathbf{Y}} p_\theta(\mathbf{X}, \mathbf{Y}) + \log p(\theta)$
$\mathcal{Q}_\mathbf{x}, \mathcal{Q}$	posterior regularization set: constrained set of desired data-conditional distributions
$\phi(\mathbf{x}, \mathbf{y})$	constraint features: used to encode posterior regularization
$\mathbf{b}$	bounds on the desired expected values of constraint features
ξ	slack variables used to allow small violations of constraints
$J_\mathcal{Q}(\theta)$	posterior regularized likelihood: $\mathcal{L}(\theta) - \mathbf{KL}(\mathcal{Q} \parallel p_\theta(\mathbf{Y}\|\mathbf{X}))$

Table 2.1: Summary of notation used.

2.2 Regularization via Posterior Constraints

The goal of the posterior regularization framework is to restrict the space of the model posteriors on unlabeled data as a way to guide the model towards desired behavior. In this section we describe a version of PR specified with respect to a set of constraints. In this case, posterior information is specified with sets $\mathcal{Q}$ of allowed distributions over the hidden variables $\mathbf{y}$. We will define $\mathcal{Q}$ in terms of *constraint* features $\phi(\mathbf{X}, \mathbf{Y})$ and their expectations. [2]

> **Running Example** *Recall that in our running example, we want to bias learning so that each sentence is labeled to contain at least one verb. To encode this formally, we define a feature* $\phi(\mathbf{x}, \mathbf{y}) =$ *"number of verbs in* $\mathbf{y}$*", and require that this feature has expectation at least 1. For consistency with the rest of the exposition and standard optimization literature, we will use the equivalent* $\phi(\mathbf{x}, \mathbf{y}) =$

[2] Note: the constraint features do not appear anywhere in the model. If the model has a log-linear form, then it would be defined with respect to a different set of *model* features, not related to the *constraint* features we consider here.

"negative number of verbs in $\mathbf{y}$" and require this has expectation at most -1:[3]

$$\mathcal{Q}_{\mathbf{x}} = \{q_{\mathbf{x}}(\mathbf{y}) \ : \ \mathbf{E}_q[\phi(\mathbf{x}, \mathbf{y})] \leq -1\}$$

Note that we enforce the constraint only in expectation, so there might be a labeling with non-zero probability that does not contain a verb. To actually enforce this constraint in the model would break the first-order Markov property of the distribution. [4] *In order to also require at least one noun per sentence in expectation, we would add another constraint feature, so that ϕ would be a function from $\mathbf{x}, \mathbf{y}$ pairs to $\mathbb{R}^2$.*

We define $\mathcal{Q}$, the set of valid distributions, with respect to the *expectations* of constraint features, rather than their probabilities, so that our objective leads to an efficient algorithm. As we will see later in this section, we also require that the constraint features decompose as a sum in order to ensure an efficient algorithm. More generally than in the running example, we will define constraints over an entire corpus:

$$\textbf{Constrained Posterior Set}: \qquad \mathcal{Q} = \{q(\mathbf{Y}) : \mathbf{E}_q[\phi(\mathbf{X}, \mathbf{Y})] \leq \mathbf{b}\}. \tag{2.1}$$

In words, $\mathcal{Q}$ denotes the region where constraint feature expectations are bounded by $\mathbf{b}$. Additionally, it is often useful to allow small violations whose norm is bounded by $\epsilon \geq 0$:

$$\textbf{Constrained Set (with slack)}: \quad \mathcal{Q} = \{q(\mathbf{Y}) : \exists \xi, \ \mathbf{E}_q[\phi(\mathbf{X}, \mathbf{Y})] - \mathbf{b} \leq \xi; \ ||\xi||_\beta \leq \epsilon\}. \tag{2.2}$$

Here ξ is a vector of slack variables and $||\cdot||_\beta$ denotes some norm. Note that the PR method we describe will only be useful if $\mathcal{Q}$ is non-empty:

Assumption 2.1. *$\mathcal{Q}$ is non-empty.*

We explore several types of constraints in Chapters 6-10, including: constraints similar to the running example, where each hidden state is constrained to appear at most once in expectation; constraints that bias two models to agree on latent variables in expectation;

[3] Note that the distribution $q_{\mathbf{x}}(\mathbf{y})$ and $\mathcal{Q}_{\mathbf{x}}$ depend on $\mathbf{x}$ because the features $\phi(\mathbf{x}, \mathbf{y})$ might depend on the particular example $\mathbf{x}$. In order to recover constraints for the entire corpus $\mathbf{X}$ we can stack the $\phi(\mathbf{x}, \mathbf{y})$ for each sentence $\mathbf{x}$ into a long vector $\phi(\mathbf{X}, \mathbf{Y})$. This corresponds to computing the intersection of the constraints $\mathcal{Q} = \bigcap_{\mathbf{x}} \mathcal{Q}_{\mathbf{x}}$.

[4] At every position in the sentence, we would need to know whether a verb was used at any other position.

constraints that enforce a particular group-sparsity of the posterior moments. The constraint set defined in Equation 2.2 is usually referred to as inequality constraints with slack, since setting $\epsilon = 0$ enforces inequality constraints strictly. The derivations for equality constraints are very similar to the derivations for inequality so we leave them out in the interest of space. Note also that we can encode equality constraints by adding two inequality constraints, although this will leave us with twice as many variables in the dual. The assumption of linearity of the constraints is computationally important, as we will show below. For now, we do not make any assumptions about the features $\phi(\mathbf{x}, \mathbf{y})$, but if they factor in the same way as the model, then we can use the same inference algorithms in PR training as we use for the original model (see Proposition 2.2). In PR, the log-likelihood of a model is penalized with the KL-divergence between the desired distribution space $\mathcal{Q}$ and the model posteriors,

$$\mathbf{KL}(\mathcal{Q} \parallel p_\theta(\mathbf{Y}|\mathbf{X})) = \min_{q \in \mathcal{Q}} \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})). \tag{2.3}$$

The posterior-regularized objective is:

$$\textbf{Posterior Regularized Likelihood}: \quad J_{\mathcal{Q}}(\theta) = \mathcal{L}(\theta) - \mathbf{KL}(\mathcal{Q} \parallel p_\theta(\mathbf{Y}|\mathbf{X})). \tag{2.4}$$

The objective trades off likelihood and distance to the desired posterior subspace (modulo getting stuck in local maxima) and provides an effective means of controlling the posteriors. In many cases, prior knowledge is easy to specify in terms of posteriors, and much more difficult to specify as priors on model parameters or by explicitly adding constraints to the model. A key advantage of using regularization on posteriors is that the learned model itself remains simple and tractable, while during learning it is driven to obey the constraints through setting appropriate parameters θ. The advantage of imposing the constraints via KL-divergence from the posteriors is that the objective above can be optimized using a simple EM scheme described in Section 2.5. It is also possible to use a similar algorithm to maximize $\mathcal{L}(\theta) - \alpha\mathbf{KL}(\mathcal{Q} \parallel p_\theta(\mathbf{Y} \mid \mathbf{X}))$, for $\alpha \in [0, 1]$. See Appendix B for details of the case when $\alpha \neq 1$. Note that the algorithm we will present in Section 2.5 will not allow us to optimize an objective with $\alpha > 1$, and this leads us to have both a KL-penalty term in Equation 2.4 and also to potentially have slack in the definition of the

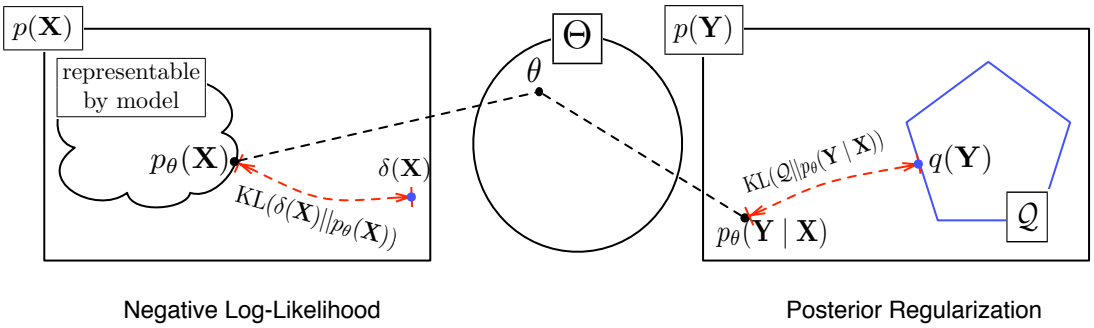


Figure 2.1: An illustration of the PR objective for generative models, as a sum of two KL terms. The symbol Θ represents the set of possible model parameters, $\delta(\mathbf{X})$ is a distribution that puts probability 1 on $\mathbf{X}$ and 0 on all other assignments. Consequently $\mathbf{KL}(\delta(\mathbf{X})||p_\theta(\mathbf{X})) = \mathcal{L}(\theta)$. (We ignore the parameter prior and additional labeled data in this figure for clarity.)

constraint set $\mathcal{Q}$. We do not need to allow slack in the objective, as long as we are sure that the constraint set $\mathcal{Q}$ is non-empty. At increased computational cost, it is also possible to eliminate the KL-penalty portion of the objective, instead directly constraining the model's posterior distribution to be inside the constraint set $p_\theta(\mathbf{Y}|\mathbf{X}) \in \mathcal{Q}$. See Section 4 for details. Figure 2.1 illustrates the objective in Equation 2.4. Normal maximum likelihood training is equivalent to minimizing the KL distance between the distribution concentrated on $\mathbf{X}$ and the set of distributions representable by the model. Any particular setting of the model parameters results in the posterior distribution $p_\theta(\mathbf{Y}|\mathbf{X})$. PR adds to the maximum likelihood objective a corresponding KL distance for this distribution. If $\mathcal{Q}$ has only one distribution, then we recover labeled maximum likelihood training. This is one of the justifications for the use and the particular direction of the KL distance in the penalty term.

> **Running Example** *In order to represent a corpus-wide constraint set $\mathcal{Q}$ for our POS problem, we stack the constraint features into a function from $\mathbf{X}, \mathbf{Y}$ pairs (sentences, part-of-speech sequences) to $\mathbb{R}^{2|\mathbf{X}|}$, where $|\mathbf{X}|$ is the number of sentences in our unlabeled corpus. For the POS tagging example, the PR objective penalizes parameters that do not assign each sentence a verb and a noun in expectation.*

For PR to be successful, the model $p_\theta(\mathbf{Y}|\mathbf{X})$ has to be expressive enough to ensure that the learned model has posteriors $p_\theta(\mathbf{Y}|\mathbf{X})$ in or nearly in $\mathcal{Q}$. However, we could also use

$q(\mathbf{Y}) = \arg\min_{q' \in \mathcal{Q}} \mathbf{KL}(q'(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X}))$ for prediction instead of $p_\theta(\mathbf{Y}|\mathbf{X})$. We will see in Sections 6 and 8 that this sometimes results in improved performance. Chang et al. [2007] report similar results for their constraint-driven learning framework.

2.3 Slack Constraints vs. Penalty

In order for our objective to be well defined, $\mathcal{Q}$ must be non-empty. When there are a large number of constraints, or when the constraint features ϕ are defined by some instance-specific process, it might not be easy to choose constraint values $\mathbf{b}$ and slack ϵ that lead to satisfiable constraints. It is sometimes easier to penalize slack variables instead of setting a bound ϵ on their norm. In these cases, we add a slack penalty to the regularized likelihood objective in Equation 2.4:

$$\begin{aligned} \mathcal{L}(\theta) \quad - \min_{q,\xi} \quad & \mathbf{KL}\left(q(\mathbf{Y}) \,||\, p_\theta(\mathbf{Y}|\mathbf{X})\right) + \sigma \left|\left|\xi\right|\right|_\beta \\ \text{s.t.} \quad & \mathbf{E}_q[\phi(\mathbf{X}, \mathbf{Y})] - \mathbf{b} \leq \xi. \end{aligned} \tag{2.5}$$

The slack-constrained and slack-penalized versions of the objectives are equivalent in the sense that they follow the same regularization path: for every ϵ there exists some σ that results in identical parameters θ. Note that while we have used a norm $||\cdot||_\beta$ to impose a cost on violations of the constraints, we could have used any arbitrary convex penalty function, for which the minimal q is easily computable. We use norms throughout the thesis because of mathematical convenience, but in general we could replace $||\cdot||_\beta$ with any convex function, subject to its efficient computability.

2.3.1 Computing the Posterior Regularizer

In this section, we describe how to compute the objective we have introduced for fixed parameters θ. The regularization term is stated in Equations 2.4 and 2.5 in terms of an optimization problem. We assume that we have algorithms to do inference[5] in the statistical model of interest, p_θ. We describe the computation of the regularization term for the

[5] Specifically, we need to be able to compute marginal distributions efficiently.

inequality constraints: [6]

$$\min_{q,\xi} \quad \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \qquad \text{s.t.} \quad \begin{array}{l} \mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b} \leq \xi; \quad ||\xi||_\beta \leq \epsilon; \\ q(\mathbf{Y}) \geq 0; \sum_{\mathbf{Y}} q(\mathbf{Y}) = 1 \end{array} \tag{2.6}$$

Proposition 2.1. *For appropriately decomposable constraint features ϕ, the regularization problems for PR with inequality constraints in Equation 2.6 can be solved efficiently in its dual form. The primal solution q^* is unique since KL divergence is strictly convex and is given in terms of the dual solution λ^* by:*

$$q^*(\mathbf{Y}) = \frac{p_\theta(\mathbf{Y}|\mathbf{X}) \exp\{-\lambda^* \cdot \phi(\mathbf{X},\mathbf{Y})\}}{Z(\lambda^*)} \tag{2.7}$$

where $Z(\lambda^*) = \sum_{\mathbf{Y}} p_\theta(\mathbf{Y}|\mathbf{X}) \exp\{-\lambda^* \cdot \phi(\mathbf{X},\mathbf{Y})\}$. Define $||\cdot||_{\beta^*}$ as the dual norm of $||\cdot||_\beta$.[7] The dual of the problem in Equation 2.6 is:

$$\max_{\lambda \geq 0} \; -\mathbf{b} \cdot \lambda - \log Z(\lambda) \; - \epsilon \, ||\lambda||_{\beta^*} . \tag{2.8}$$

The proof is included in Appendix C using standard Lagrangian duality results and strict convexity of KL (e.g., Bertsekas [1999]). The dual form in Equation 2.8 is typically computationally more tractable than the primal form (Equation 2.6) because there is one dual variable per expectation constraint, while there is one primal variable per labeling $\mathbf{Y}$. For structured models, this is typically intractable. An analogous proposition can be proven for the objective with penalties (Equation 2.5), with almost identical proof. We omit this for brevity.

2.4 Factored $q(\mathbf{Y})$ for Factored Constraints

The form of the optimal q with respect to $p_\theta(\mathbf{Y}|\mathbf{X})$ and ϕ has important computational implications.

[6] For simplicity of notation we will implicitly assume the constraint that q is a distribution in the sequel.

[7] The dual of a norm $||\cdot||_\beta$ is defined as $||\xi||_{\beta^*} = \max_\alpha \xi \cdot \alpha \, \text{s.t.} \, ||\alpha||_\beta \leq 1$.

Proposition 2.2. *If $p_\theta(\mathbf{Y}|\mathbf{X})$ factors as a product of clique potentials over a set of cliques $\mathcal{C}$, and $\phi(\mathbf{X}, \mathbf{Y})$ factors as a sum over some subset of those cliques, then the optimizer $q^*(\mathbf{Y})$ of Equation 2.7 will also factor as a product of potentials of cliques in $\mathcal{C}$.*

This is easy to show. Our assumptions are a factorization for p_θ:

$$\textbf{Factored Posteriors}: \quad p(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_{c \in \mathcal{C}} \psi(\mathbf{X}, \mathbf{Y}_c) \tag{2.9}$$

and the same factorization for ϕ:

$$\textbf{Factored Features}: \quad \phi(\mathbf{X}, \mathbf{Y}) = \sum_{c \in \mathcal{C}} \phi(\mathbf{X}, \mathbf{Y}_c) \tag{2.10}$$

which imply that $q^*(\mathbf{Y})$ will also factor as a product over the cliques $\mathcal{C}$:

$$\begin{aligned} \textbf{Factored Solution}: \quad q^*(\mathbf{Y}) &= \frac{1}{Z(\mathbf{X})Z(\lambda^*)} \prod_{c \in \mathcal{C}} \psi(\mathbf{X}, \mathbf{Y}_c) \ \exp\{-\lambda^* \cdot \phi(\mathbf{X}, \mathbf{Y}_c)\} \\ &= \frac{1}{Z'(\mathbf{X})} \prod_{c \in \mathcal{C}} \psi'(\mathbf{X}, \mathbf{Y}_c), \end{aligned} \tag{2.11}$$

where $\psi'(\mathbf{X}, \mathbf{Y}_c) = \psi(\mathbf{X}, \mathbf{Y}_c) \exp\{-\lambda^* \cdot \phi(\mathbf{X}, \mathbf{Y}_c)\}$ and $Z'(\mathbf{X}) = Z(\mathbf{X})Z(\lambda^*)$.

2.5 Generative Posterior Regularization via Expectation Maximization

This section presents an optimization algorithm for the PR objective. The algorithm we present is a minorization-maximization algorithm akin to EM, and both slack-constrained and slack-penalized formulations can be optimized using it. To simplify the exposition, we focus first on slack-constrained version, and leave a treatment of optimization of the slack-penalized version to Section 2.6.

Recall the standard expectation maximization (EM) algorithm used to optimize marginal likelihood $\mathcal{L}(\theta) = \log \sum_{\mathbf{Y}} p_\theta(\mathbf{X}, \mathbf{Y})$. Again, for clarity of exposition, we ignore $\log p(\theta)$, the prior on θ, as well as $\log p_\theta(\mathbf{X}_L, \mathbf{Y}_L)$, the labeled data term, as they are

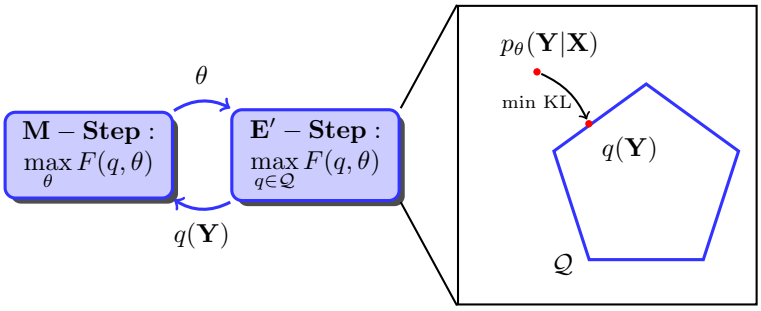


Figure 2.2: Modified EM for optimizing generative PR objective $\mathcal{L}(\theta) - \mathbf{KL}(\mathcal{Q} \parallel p_\theta(\mathbf{Y}|\mathbf{X}))$.

simple to incorporate, just as in regular EM. Neal and Hinton [1998] describe an interpretation of the EM algorithm as block coordinate ascent on a function that lower-bounds $\mathcal{L}(\theta)$, which we also use below. By Jensen's inequality, we define a lower-bound $F(q,\theta)$ as

$$\mathcal{L}(\theta) = \log \sum_{\mathbf{Y}} q(\mathbf{Y}) \frac{p_\theta(\mathbf{X},\mathbf{Y})}{q(\mathbf{Y})} \geq \sum_{\mathbf{Y}} q(\mathbf{Y}) \log \frac{p_\theta(\mathbf{X},\mathbf{Y})}{q(\mathbf{Y})} = F(q,\theta). \tag{2.12}$$

we can re-write $F(q,\theta)$ as

$$\begin{aligned} F(q,\theta) &= \sum_{\mathbf{Y}} q(\mathbf{Y}) \log(p_\theta(\mathbf{X}) p_\theta(\mathbf{Y}|\mathbf{X})) - \sum_{\mathbf{Y}} q(\mathbf{Y}) \log q(\mathbf{Y}) \\ &= \mathcal{L}(\theta) - \sum_{\mathbf{Y}} q(\mathbf{Y}) \log \frac{q(\mathbf{Y})}{p_\theta(\mathbf{Y}|\mathbf{X})} \\ &= \mathcal{L}(\theta) - \mathbf{KL}(q(\mathbf{Y}) || p_\theta(\mathbf{Y}|\mathbf{X})) \end{aligned} \tag{2.13}$$

Using this interpretation, we can view EM as performing coordinate ascent on $F(q,\theta)$. Starting from an initial parameter estimate θ^0, the algorithm iterates two block-coordinate ascent steps until a convergence criterion is reached:

$$\mathbf{E}: q^{t+1} = \arg\max_{q} F(q,\theta^t) = \arg\min_{q} \mathbf{KL}(q(\mathbf{Y}) \parallel p_{\theta^t}(\mathbf{Y} \mid \mathbf{X})) \tag{2.14}$$

$$\mathbf{M}: \theta^{t+1} = \arg\max_{\theta} F(q^{t+1},\theta) = \arg\max_{\theta} \mathbf{E}_{q^{t+1}} [\log p_\theta(\mathbf{X},\mathbf{Y})] \tag{2.15}$$

where the minimization in Equation 2.14 is over the full set of distributions of hidden variables $\mathbf{Y}$. It is easy to see that the E-step sets $q^{t+1}(\mathbf{Y}) = p_{\theta^t}(\mathbf{Y}|\mathbf{X})$.

The PR objective (Equation 2.4) is

$$J_{\mathcal{Q}}(\theta) = \max_{q \in \mathcal{Q}} F(q, \theta) = \mathcal{L}(\theta) - \min_{q \in \mathcal{Q}} \mathbf{KL}(q(\mathbf{Y}) || p_\theta(\mathbf{Y}|\mathbf{X})), \tag{2.16}$$

where $\mathcal{Q} = \{q(\mathbf{Y}) : \exists \xi, \quad \mathbf{E}_q[\phi(\mathbf{X}, \mathbf{Y})] - \mathbf{b} \leq \xi; \ ||\xi||_\beta \leq \epsilon\}$. In order to optimize this objective, it suffices to modify the E-step to include the constraints:

$$\mathbf{E}' : q^{t+1} = \arg\max_{q \in \mathcal{Q}} F(q, \theta^t) = \arg\min_{q \in \mathcal{Q}} \mathbf{KL}(q(\mathbf{Y}) \parallel p_{\theta^t}(\mathbf{Y}|\mathbf{X})) \tag{2.17}$$

The projected posteriors $q^{t+1}(\mathbf{Y})$ are then used to compute sufficient statistics and update the model's parameters in the M-step, which remains unchanged, as in Equation 2.15. This scheme is illustrated in Figure 2.2.

Proposition 2.3. *The modified EM algorithm illustrated in Figure 2.2, which iterates the modified E-step (Equation 2.17) with the normal M-step (Equation 2.15), monotonically increases the PR objective:* $J_{\mathcal{Q}}(\theta^{t+1}) \geq J_{\mathcal{Q}}(\theta^t)$.

Proof: The proof is analogous to the proof of monotonic increase of the standard EM objective. Essentially,

$$J_{\mathcal{Q}}(\theta^{t+1}) = F(q^{t+2}, \theta^{t+1}) \geq F(q^{t+1}, \theta^{t+1}) \geq F(q^{t+1}, \theta^t) = J_{\mathcal{Q}}(\theta^t).$$

The two inequalities are ensured by the $\mathbf{E}'$-step and M-step. $\mathbf{E}'$-step sets $q^{t+1} = \arg\max_{q \in \mathcal{Q}} F(q, \theta^t)$, hence $J_{\mathcal{Q}}(\theta^t) = F(q^{t+1}, \theta^t)$. The M-step sets $\theta^{t+1} = \arg\max_\theta F(q^{t+1}, \theta)$, hence $F(q^{t+1}, \theta^{t+1}) \geq F(q^{t+1}, \theta^t)$. Finally, $J_{\mathcal{Q}}(\theta^{t+1}) = \max_{q \in \mathcal{Q}} F(q, \theta^{t+1}) \geq F(q^{t+1}, \theta^{t+1})$ ■

Note that the proposition is only meaningful when $\mathcal{Q}$ is non-empty and $J_{\mathcal{Q}}$ is well-defined. As for standard EM, to prove that coordinate ascent on $F(q, \theta)$ converges to stationary points of $J_{\mathcal{Q}}(\theta)$, we need to make additional assumptions on the regularity of the likelihood function and boundedness of the parameter space as in Tseng [2004]. This analysis can be easily extended to our setting, but is beyond the scope of the current dissertation.

We can use the dual formulation of Proposition 2.1 to perform the projection. Proposition 2.2 implies that we can use the same algorithms to perform inference in our projected model q as we did in our original model p_θ. We illustrate this with the running example.

Running Example *For the POS tagging example with zero slack, the optimization problem we need to solve is:*

$$\arg\min_{q} \quad \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \qquad \text{s.t.} \quad \mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] \leq -\mathbf{1}$$

where $\mathbf{1}$ *is a vector of with 1 in each entry. The dual formulation is given by*

$$\arg\max_{\lambda \geq 0} \ \mathbf{1}\cdot\lambda - \log Z(\lambda) \quad \textit{with} \quad q^*(\mathbf{Y}) = \frac{p_\theta(\mathbf{Y}|\mathbf{X})\exp\{-\lambda^*\cdot\phi(\mathbf{X},\mathbf{Y})\}}{Z(\lambda^*)}. \tag{2.18}$$

We can solve the dual optimization problem by projected gradient ascent. The HMM model can be factored as products over sentences, and each sentence as a product of emission probabilities and transition probabilities.

$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \frac{\prod_{i=1}^{|\mathbf{x}|} p_\theta(y_i|y_{i-1})p_\theta(x_i|y_i)}{p_\theta(\mathbf{x})} \tag{2.19}$$

where $p_\theta(y_1|y_0) = p_\theta(y_1)$ *are the initial probabilities of our HMM. The constraint features* ϕ *can be represented as a sum over sentences and further as a sum over positions in the sentence:*

$$\phi(\mathbf{x},\mathbf{y}) \quad = \quad \sum_{i=1}^{|\mathbf{x}|}\phi_i(\mathbf{x},y_i) \quad = \quad \sum_{i=1}^{|\mathbf{x}|}\begin{cases}(-1,0)^\top & \textit{if } y_i \textit{ is a verb in sentence } \mathbf{x}\\ (0,-1)^\top & \textit{if } y_i \textit{ is a noun in sentence } \mathbf{x}\\ (0,0)^\top & \textit{otherwise}\end{cases} \tag{2.20}$$

combining the factored Equations 2.19 and 2.20 with the definition of $q(\mathbf{Y})$ *we see that* $q(\mathbf{Y})$ *must also factor as a first-order Markov model for each sentence.*

$$q^*(\mathbf{Y}) \propto \quad \prod_{\mathbf{x}\in\mathbf{X}}\prod_{i=1}^{|\mathbf{x}|} p_\theta(y_i|y_{i-1})p_\theta(x_i|y_i)e^{-\lambda^*\cdot\phi_i(\mathbf{x},y_i)} \tag{2.21}$$

Hence $q^*(\mathbf{Y})$ *is just a first-order Markov model for each sentence, and we can compute the normalizer* $Z(\lambda^*)$ *and marginals* $q(y_i)$ *for each example using forward-backward. This allows computation of the dual objective in Equation 2.18 as well as its gradient efficiently. The gradient of the dual objective is* $1 - \mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})]$*. We can use projected gradient [Bertsekas, 1999] to perform the optimization, and the projection can be done sentence-by-sentence allowing for online optimization*

such as stochastic gradient. Optimization for non-zero slack case can be done using projected subgradient (since the norm is not smooth).

Note that on *unseen* unlabeled data, the learned parameters θ might not satisfy the constraints on posteriors exactly, although typically they are fairly close if the model has enough capacity.

2.6 Penalized Slack via Expectation Maximization

If our objective is specified using slack-penalty such as in Equation 2.5, then we need a slightly different E-step. Instead of restricting $q \in \mathcal{Q}$, the modified $\mathbf{E}'$-step adds a cost for violating the constraints

$$\mathbf{E}' : \min_{q,\xi} \quad \mathbf{KL}\left(q(\mathbf{Y}) \,||\, p_\theta(\mathbf{Y}|\mathbf{X})\right) + \sigma \left|\left|\xi\right|\right|_\beta \quad \text{s.t.} \quad \mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b} \leq \xi \tag{2.22}$$

An analogous monotonic improvement of modified EM can be shown for the slack-penalized objective. The dual of Equation 2.22 is

$$\max_{\lambda \geq 0} \quad -\mathbf{b}\cdot\lambda - \log Z(\lambda) \qquad \text{s.t.} \quad \left|\left|\lambda\right|\right|_{\beta^*} \leq \sigma. \tag{2.23}$$

2.7 PR for Discriminative Models

The PR framework can be used to guide learning in discriminative models as well as generative models. In the case of a discriminative model, we only have $p_\theta(\mathbf{y}|\mathbf{x})$, and the likelihood does not depend on unlabeled data. Specifically,

$$\mathcal{L}^D(\theta) = \log p_\theta(\mathbf{Y}_L|\mathbf{X}_L) + \log p(\theta), \tag{2.24}$$

where $(\mathbf{Y}_L, \mathbf{X}_L)$ are any available labeled data and $\log p(\theta)$ is a prior on the model parameters. With this definition of $\mathcal{L}(\theta)$ for discriminative models we will optimize the discriminative PR objective (zero-slack case):

$$\textbf{Discriminative PR Likelihood}: \quad J^D_{\mathcal{Q}}(\theta) = \mathcal{L}^D(\theta) - \mathbf{KL}(\mathcal{Q} \parallel p_\theta(\mathbf{Y}|\mathbf{X})). \tag{2.25}$$

In the absence of both labeled data and a prior on parameters $p(\theta)$, the objective in Equation 2.4 is optimized (equal to zero) for any $p_\theta(\mathbf{Y} \mid \mathbf{X}) \in \mathcal{Q}$. If we employ a parametric prior on θ, then we will prefer parameters that come close to satisfying the constraints, where proximity is measured by KL-divergence.

Running Example *For the POS tagging example, our discriminative model might be a first order conditional random field. In this case we model:*

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{\exp\{\theta \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})\}}{Z_\theta(\mathbf{x})} \tag{2.26}$$

where $Z_\theta(\mathbf{x}) = \sum_{\mathbf{y}} \exp\{\theta \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})\}$ *is a normalization constant and* $\mathbf{f}(\mathbf{x}, \mathbf{y})$ *are the* model *features. We will use the same constraint features as in the generative case:* $\phi(\mathbf{x}, \mathbf{y}) =$ *"negative number of verbs in* $\mathbf{y}$*", and define* $\mathcal{Q}_\mathbf{x}$ *and* $q_\mathbf{x}$ *also as before. Note that* $\mathbf{f}$ *are features used to define the model and do not appear anywhere in the constraints while* ϕ *are constraint features that do not appear anywhere in the model.*

Traditionally, the EM algorithm is used for learning generative models (the model can condition on a subset of observed variables, but it must define a distribution over some observed variables). The reason for this is that EM optimizes marginal log-likelihood ($\mathcal{L}$ in our notation) of the observed data $\mathbf{X}$ according to the model. In the case of a discriminative model, $p_\theta(\mathbf{Y}|\mathbf{X})$, we do not model the distribution of the observed data, the value of $\mathcal{L}^D$ as a function of θ depends only on the parametric prior $p(\theta)$ and the labeled data. By contrast, the PR objective uses the KL term and the corresponding constraints to bias the model parameters. These constraints depend on the observed data $\mathbf{X}$ and if they are sufficiently rich and informative, they can be used to train a discriminative model. In the extreme case, consider a constraint set $\mathcal{Q}$ that contains only a single distribution q, with $q(\mathbf{Y}^*) = 1$. So, q is concentrated on a particular labeling $\mathbf{Y}^*$. In this case, the PR objective in Equation 2.25 reduces to

$$J^D_\mathcal{Q}(\theta) = \mathcal{L}^D(\theta) + \log p_\theta(\mathbf{Y}^*|\mathbf{X}) = \log p(\theta) + \log p_\theta(\mathbf{Y}_L|\mathbf{X}_L) + \log p_\theta(\mathbf{Y}^*|\mathbf{X}) \tag{2.27}$$

Thus, if $\mathcal{Q}$ is informative enough to uniquely specify a labeling of the unlabeled data, the PR objective reduces to the supervised likelihood objective. When $\mathcal{Q}$ specifies a range of

distributions, such as the one for multi view learning (Section 7), PR biases the discriminative model to have $p_\theta(\mathbf{Y}|\mathbf{X})$ close to $\mathcal{Q}$.

Equation 2.25 can also be optimized with a block-coordinate ascent, leading to an EM style algorithm very similar to the one presented in Section 2.5. We define a lower bounding function:

$$F'(q,\theta) = -\mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) = \sum_{\mathbf{Y}} q(\mathbf{Y}) \log \frac{p_\theta(\mathbf{Y}|\mathbf{X})}{q(\mathbf{Y})}. \tag{2.28}$$

Clearly, $\max_{q\in\mathcal{Q}} F'(q,\theta) = -\mathbf{KL}(\mathcal{Q} \parallel p_\theta(\mathbf{Y}|\mathbf{X}))$ so $F'(q,\theta) \leq -\mathbf{KL}(\mathcal{Q} \parallel p_\theta(\mathbf{Y}|\mathbf{X}))$ for $q \in \mathcal{Q}$.

The modified $\mathbf{E}'$ and $\mathbf{M}'$ steps are: [8]

$$\mathbf{E}' : q^{t+1} = \arg\max_{q\in\mathcal{Q}} F'(q,\theta^t) = \arg\min_{q\in\mathcal{Q}} \mathbf{KL}(q(\mathbf{Y}) \parallel p_{\theta^t}(\mathbf{Y}|\mathbf{X})) \tag{2.29}$$

$$\mathbf{M}' : \theta^{t+1} = \arg\max_{\theta} F'(q^{t+1},\theta) = \arg\max_{\theta} \mathbf{E}_{q^{t+1}} \left[\log p_\theta(\mathbf{Y}|\mathbf{X})\right]. \tag{2.30}$$

Here the difference between Equation 2.15 and Equation 2.30 is that now there is no generative component in the lower-bound $F'(q,\theta)$ and hence we have a discriminative update to the model parameters in Equation 2.30.

[8] As with the **M-step** in Equation 2.15 we have ignored the prior $p(\theta)$ on model parameters and the labeled data terms, which can be easily incorporated in the $\mathbf{M}'$ step.

Chapter 3

Summary of Applications

Because the PR framework allows very diverse prior information to be specified in a single formalism, the application Chapters (§6-§10) are very diverse in nature. This section attempts to summarize their similarities and differences without getting into the details of the problem applications and intuition behind the constraints. Table 3.1 summarizes the applications and constraints described in the rest of the document while Table 3.2 summarizes the meanings of the variables $\mathbf{x}$, $\mathbf{y}$ and $\phi(\mathbf{X}, \mathbf{Y})$ as well as the optimization procedures used for the applications presented in the sequel.

In the statistical word alignment application described in Chapter 6, the goal is to identify pairs or sets of words that are direct translations of each other. The statistical models

§#	**Problem**	**Gen/Disc**	p/q	**Summary of Structural Constraints**
§6	Word Alignment	G	q	Translation process is symmetric and bijective
§7	Multi-view learning	D	q	Multiple views should agree on label distribution
§8	Dependency Parsing	G+D	p	Noisy, partially observed labels encoded in ϕ and $\mathbf{b}$
§9	Part-of-speech induction	G	p	Sparsity structure independent of model parameters: each word should be generated by a small number of POS tags
§10	Grammar induction	p	G	Sparsity structure independent of model parameters: the number parse edge types (grammar rules) should be small

Table 3.1: Summary of applications of Posterior Regularization described in this dissertation. Gen/Disc refers to generative or discriminative models. The p/q column shows whether we use the original model p or the projected distribution q at decode time.

Application	Symbol	Meaning
Word Alignment	$\mathbf{x}$	Pair of sentences that are translations of each other.
	$\mathbf{y}$	Set of alignment links between words in the sentences (exponentially many possibilities).
	$\phi(\mathbf{x}, \mathbf{y})$	Bijective: number of times each source word is used in the alignment $\mathbf{y}$. Symmetric: expected difference in number of times each link is used by source→target model and target →source model.
	OPT	Bijective: projected gradient. Symmetric: L-BFGS.
Multi-view learning	$\mathbf{x}$	Varies by application.
	$\mathbf{y}$	Varies by application.
	$\phi(\mathbf{x}, \mathbf{y})$	Indexed by label; ± 1 if only one model predicts the label, and 0 if both or none predict the label.
	OPT	Closed form.
Dependency Parsing	$\mathbf{x}$	Natural language sentence as a sequence of words.
	$\mathbf{y}$	Dependency parse tree (set of edges from one word to another, forming a tree).
	$\phi(\mathbf{x}, \mathbf{y})$	Number of edges in $\mathbf{y}$ that are in the set of translated edges.
	OPT	Line search.
Part-of-speech induction	$\mathbf{x}$	Natural language sentence as a sequence of words.
	$\mathbf{y}$	Sequence of syntactic categories, one for each word.
	$\phi(\mathbf{X}, \mathbf{Y})$	Indexed by w, i, s; 1 if the i^{th} occurrence of word w in the corpus is tagged with syntactic category s, and 0 otherwise.
	OPT	Projected gradient.

Table 3.2: Summary of input and output variable meanings as well as meanings of constraint features and optimization methods used (OPT) for the applications summarized in Table 3.1.

used suffer from what is known as a garbage collector effect: the likelihood function of the simplistic translation models used prefers to align sections that are not literal translations to rare words, rather than leaving them unaligned [Moore, 2004]. This results in each rare word in a source language being aligned to 4 or 5 words in the target language. To alleviate this problem, we introduce constraint features that count how many target words are aligned to each source word, and use PR to encourage models where this number is small in expectation. Modifying the model itself to include such a preference would break independence and make it intractable.

The multi-view learning application described in Chapter 7 leverages two or more sources of input ("views") along with unlabeled data. The requirement is to train two

models, one for each view, such that they usually agree on the labeling of the unlabeled data. We can do this using PR, and we recover the Bhattacharyya distance as a regularizer. The PR approach also extends naturally to structured problems, and cases where we only want partial agreement.

The grammar induction application of Chapter 8 takes advantage of an existing parser for a resource-rich language to train a comparable resource for a resource-poor language. Because the two languages have different syntactic structures, and errors in word alignment abound, using such out-of-language information requires the ability to train with missing and noisy labeling. This is achieved using PR constraints that guide learning to prefer models that tend to agree with the noisy labeling wherever it is provided, while standard regularization guides learning to be self-consistent.

Finally, Chapters 9 and 10 describes an application of PR to ensure a particular sparsity structure, which can be independent of the structure of the model. Chapter 9 focuses on the problem of unsupervised part-of-speech induction, where we are given a sample of text and are required to specify a syntactic category for each token in the text. A well-known but difficult to capture piece of prior knowledge for this problem is that each word type should only occur with a small number of syntactic categories, even though there are some syntactic categories that occur with many different word types. By using an ℓ_1/ℓ_∞ norm on constraint features we are able to encourage the model to have precisely this kind of sparsity structure, and greatly increase agreement with human-generated syntactic categories. Chapter 10 extends the application of the ℓ_1/ℓ_∞ norm to the problem of grammar induction. In the case of grammar induction, we view edges as belonging to different types, based on the parent part-of-speech and child part-of-speech. The idea we want to encode is that only a small number of the possible types should occur in the language. In the setting of hard grammars, this amounts to saying that the grammar should be small.

Table 3.1 also shows for each application whether we use the distribution over hidden variables given by the model parameters $p_\theta(\mathbf{Y}|\mathbf{X})$ to decode, or whether we first project the distribution to the constraint set and use $q(\mathbf{Y})$ to decode. In general we found that when applying the constraints on the labeled data is sensible, performing the projection before decoding tends to improve performance. For the word alignment application and the multi-

view learning application we found decoding with the projected distribution improved performance. By contrast, for dependency parsing, we do not have the English translations at test time and so we cannot perform a projection. For part-of-speech induction the constraints are over the entire corpus, and different regularization strengths might be needed for the training and test sets. Since we did not want to tune a second hyperparameter, we instead decoded with p.

Chapter 4

Related Frameworks

This chapter is largely based on Ganchev et al. [2010]. The work related to learning with constraints on posterior distributions is described in chronological order in the following three sections. An overall summary is most easily understood in reverse chronological order though, so we begin with a few sentences detailing the connections to it in that order. Liang et al. [2009] describe how we can view constraints on posterior distributions as measurements in a Bayesian setting, and note that inference using such information is intractable. By approximating this problem, we recover either the generalized expectation constraints framework of Mann and McCallum [2007], or with a further approximation we recover a special case of the posterior regularization framework presented in Chapter 2. Finally, a different approximation recovers the constraint driven learning framework of Chang et al. [2007]. To the best of our knowledge, we are the first to describe all these connections.

4.1 Constraint Driven Learning

Chang et al. [2007, 2008] describe a framework called constraint driven learning (CODL) that can be viewed as an approximation to optimizing the slack-penalized version of the PR objective (Equation 2.5). Chang et al. [2007] are motivated by hard-EM, where the distribution q is approximated by a single sample at the mode of $\log p_\theta(\mathbf{Y}|\mathbf{X})$. Chang et al. [2007] propose to augment $\log p_\theta(\mathbf{Y}|\mathbf{X})$ by adding to it a penalty term based on

some domain knowledge. When the penalty terms are well-behaved, we can view them as adding a cost for violating expectations of constraint features ϕ. In such a case, CODL can be viewed as a "hard" approximation to the PR objective:

$$\arg\max_{\theta} \ \mathcal{L}(\theta) \quad - \min_{q \in M} \Big(\mathbf{KL}(q(\mathbf{Y}) || p_\theta(\mathbf{Y}|\mathbf{X})) \ + \ \sigma \, ||\mathbf{E}_q[\phi(\mathbf{X}, \mathbf{Y})] - \mathbf{b}||_\beta \Big) \tag{4.1}$$

where M is the set of distributions concentrated on a single $\mathbf{Y}$. The modified E-Step becomes:

$$\text{CODL } \mathbf{E}'\text{-step}: \ \max_{\mathbf{Y}} \log p_\theta(\mathbf{Y}|\mathbf{X}) \ - \ \sigma \, ||\phi(\mathbf{X}, \mathbf{Y}) - \mathbf{b}||_\beta \tag{4.2}$$

Because the constraints used by Chang et al. [2007] do not allow tractable inference, they use a beam search procedure to optimize the min-KL problem. Additionally they consider a K-best variant where instead of restricting themselves to a single point estimate for q, they use a uniform distribution over the top K samples.

Carlson et al. [2010] train several named entity and relation extractors concurrently in order to satisfy type constraints and mutual exclusion constraints. Their algorithm is related to CODL in that hard assignments are made in a way that guarantees the constraints are satisfied. However, their algorithm is motivated by adding these constraints to algorithms that learn pattern extractors: at each iteration, they make assignments only to the highest confidence entities, which are then used to extract high confidence patterns for use in subsequent iterations. By contrast hard EM and CODL would make assignments to every instance and change these assignments over time. Daumé III [2008] also use constraints to filter out examples for self-training and also do not change the labels.

4.2 Generalized Expectation Criteria

Generalized expectation criteria (GE) allow a user to specify preferences about model expectations in the form of linear constraints on some feature expectations [Mann and McCallum, 2007, 2008, 2010]. As with PR, a set of constraint features ϕ are introduced, and a penalty term is added to the log-likelihood objective. The GE objective is

$$\max_{\theta} \ \mathcal{L}(\theta) - \sigma \, ||\mathbf{E}_{p_\theta}[\phi(\mathbf{X}, \mathbf{Y})] - \mathbf{b}||_\beta \, . \tag{4.3}$$

where $||\cdot||_\beta$ is typically the l_2 norm (Druck et al. [2009] use l_2^2) or a distance based on KL divergence [Mann and McCallum, 2008], and the model is a log-linear model such as maximum entropy or a CRF.

The idea leading to this objective is the following: Suppose that we only had enough resources to make a very small number of measurements when collecting statistics about the true distribution $p^*(\mathbf{y}|\mathbf{x})$. If we try to create a maximum entropy model using these statistics we will end up with a very impoverished model. It will only use a small number of features and consequently will fail to generalize to instances where these features cannot occur. In order to train a more feature-rich model, GE defines a wider set of *model* features $\mathbf{f}$ and uses the small number of estimates based on constraint features ϕ to guide learning. By using l_2 regularization on model parameters, we can ensure that a richer set of model features are used to explain the desired expectations.

Druck et al. [2009] use a gradient ascent method to optimize the objective. Unfortunately, because the second term in the GE objective (Equation 4.3) couples the constraint features ϕ and the model parameters θ, the gradient requires computing the covariance between model features $\mathbf{f}$ and the constraint features ϕ under p_θ. Specifically, for log-linear models, this derivative is the covariance of the constraint features with the model features:

$$\frac{\partial \mathbf{E}_{p_\theta}[\phi(\mathbf{X},\mathbf{Y})]}{\partial \theta} = \mathbf{E}_{p_\theta}[\mathbf{f}(\mathbf{X},\mathbf{Y})\phi(\mathbf{X},\mathbf{Y})] - \mathbf{E}_{p_\theta}[\phi(\mathbf{X},\mathbf{Y})]\mathbf{E}_{p_\theta}[\mathbf{f}(\mathbf{X},\mathbf{Y})] \tag{4.4}$$

Because of this coupling, the complexity of the dynamic program needed to compute the gradient is higher than the complexity of the dynamic program for the model. In the case of graphical models where $\mathbf{f}$ and ϕ have the same Markov dependencies, computing this gradient usually squares the running time of the dynamic program. A more efficient dynamic program might be possible [Li and Eisner, 2009, Pauls et al., 2009], however current implementations are prohibitively slower than PR when there are many constraint features.

In order to avoid the costly optimization procedure described above, Bellare et al. [2009] propose a variational approximation. Recall that at a high level, the difficulty in optimizing Equation 4.3 is because the last term couples the constraint features ϕ with the model parameters θ. In order to separate out these quantities, Bellare et al. [2009] introduce an auxiliary distribution $q(\mathbf{Y}) \approx p_\theta(\mathbf{Y}|\mathbf{X})$, which is used to approximate the last term in

Equation 4.3. The variational objective contains three terms instead of two:

$$\arg\max_{\theta} \ \mathcal{L}(\theta) \quad - \min_{q} \Big(\mathbf{KL}(q(\mathbf{Y})||p_\theta(\mathbf{Y}|\mathbf{X})) \ + \ \sigma \, ||\mathbf{E}_q[\phi(\mathbf{X}, \mathbf{Y})] - \mathbf{b}||_\beta \Big) . \quad (4.5)$$

This formulation is identical to the slack-penalized version of PR, and Bellare et al. [2009] use the same optimization procedure (described in Chapter 2). Because both the minorization and the maximization steps implement minimum Kullback-Leibler projections, Bellare et al. [2009] refer to this algorithm as alternating projections. Note that PR can also be trained in an online fashion, and Ganchev et al. [2009a] use an online optimization for this objective to train a dependency parser. These experiments are described in Chapter 8.

Closely related to GE, is the work of Quadrianto et al. [2009]. The authors describe a setting where the constraint values, $\mathbf{b}$, are chosen as the empirical estimates on some labeled data. They then train a model to have high likelihood on the labeled data, but also match the constraint features on unlabeled data. They show that for appropriately chosen constraint features, the estimated constraint values should be close to the true means, and show good experimental improvements on an image retrieval task.

4.3 Measurements in a Bayesian Framework

Liang et al. [2009] approach the problem of incorporating prior information about model posteriors from a Bayesian point of view. They motivate their approach using the following caricature. Suppose we have log-linear model $p_\theta(\mathbf{y}|\mathbf{x}) \propto \exp(\theta \cdot \mathbf{f}(\mathbf{y}, \mathbf{x}))$. In addition to any labeled data $(\mathbf{X}_L, \mathbf{Y}_L)$, we also have performed some additional experiments[1]. In particular, we have observed the expected values of some constraint features $\phi(\mathbf{X}, \mathbf{Y})$ on some unlabeled data $\mathbf{X}$. Because there is error in measurement, they observe $\mathbf{b} \approx \phi(\mathbf{X}, \mathbf{Y})$. Figure 4.1 illustrates this setting. The leftmost nodes represent $(\mathbf{x}, \mathbf{y})$ pairs from the labeled data $(\mathbf{X}_L, \mathbf{Y}_L)$. The nodes directly to the right of θ represent unlabeled $(\mathbf{x}, \mathbf{y})$ pairs from the unlabeled data $\mathbf{X}$. All the data are tied together by the dependence on the model parameters θ. The constraint features take as input the unlabeled data set $\mathbf{X}$ as well as a full labeling $\mathbf{Y}$,

[1] In their exposition, Liang et al. [2009] incorporate labeled data by including the labels among experiments. We prefer to separate these types of observations because full label observations do not require approximations.

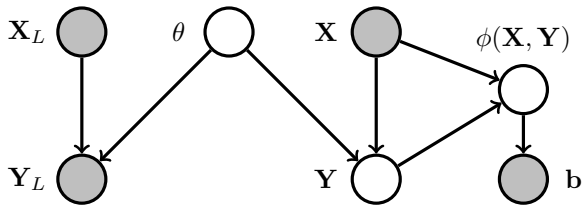


Figure 4.1: The model used by Liang et al. [2009], using our notation. We have separated treatment of the labeled data $(\mathbf{X}_L, \mathbf{Y}_L)$ from treatment of the unlabeled data $\mathbf{X}$.

and produce some value $\phi(\mathbf{X}, \mathbf{Y})$, which is never observed directly. Instead, we observe some noisy version $\mathbf{b} \approx \phi(\mathbf{X}, \mathbf{Y})$. The measured values $\mathbf{b}$ are distributed according to some noise model $p_N(\mathbf{b}|\phi(\mathbf{X}, \mathbf{Y}))$. Liang et al. [2009] note that the optimization is convex for log-concave noise and use box noise in their experiments, giving $\mathbf{b}$ uniform probability in some range near $\phi(\mathbf{X}, \mathbf{Y})$.

In the Bayesian setting, the model parameters θ as well as the observed measurement values $\mathbf{b}$ are random variables. Liang et al. [2009] use the mode of $p(\theta|\mathbf{X}_L, \mathbf{Y}_L, \mathbf{X}, \mathbf{b})$ as a point estimate for θ:

$$\arg\max_{\theta} \; p(\theta|\mathbf{X}_L, \mathbf{Y}_L, \mathbf{X}, \mathbf{b}) \quad = \quad \arg\max_{\theta} \; \sum_{\mathbf{Y}} p(\theta, \mathbf{Y}, \mathbf{b}|\mathbf{X}, \mathbf{X}_L, \mathbf{Y}_L), \tag{4.6}$$

with equality because $p(\theta|\mathbf{X}_L, \mathbf{Y}_L, \mathbf{X}, \mathbf{b}) \propto p(\theta, \mathbf{b}|\mathbf{X}_L, \mathbf{Y}_L, \mathbf{X}) = \sum_{\mathbf{Y}} p(\theta, \mathbf{Y}, \mathbf{b}|\mathbf{X}, \mathbf{X}_L, \mathbf{Y}_L)$. Liang et al. [2009] focus on computing $p(\theta, \mathbf{Y}, \mathbf{b}|\mathbf{X}, \mathbf{X}_L, \mathbf{Y}_L)$. They define their model for this quantity as follows:

$$p(\theta, \mathbf{Y}, \mathbf{b}|\mathbf{X}, \mathbf{X}_L, \mathbf{Y}_L) \quad = \quad p(\theta|\mathbf{X}_L, \mathbf{Y}_L) \; p_\theta(\mathbf{Y}|\mathbf{X}) \; p_N(\mathbf{b}|\phi(\mathbf{X}, \mathbf{Y})) \tag{4.7}$$

where the $\mathbf{Y}$ and $\mathbf{X}$ are particular instantiations of the random variables in the entire unlabeled corpus $\mathbf{X}$. Equation 4.7 is a product of three terms: a prior on θ, the model probability $p_\theta(\mathbf{Y}|\mathbf{X})$, and a noise model $p_N(\mathbf{b}|\phi)$. The noise model is the probability that we observe a value, $\mathbf{b}$, of the measurement features ϕ, given that its actual value was $\phi(\mathbf{X}, \mathbf{Y})$. The idea is that we model errors in the estimation of the posterior probabilities as noise in the measurement process. Liang et al. [2009] use a uniform distribution over $\phi(\mathbf{X}, \mathbf{Y}) \pm \epsilon$, which they call "box noise". Under this model, observing $\mathbf{b}$ farther than ϵ from $\phi(\mathbf{X}, \mathbf{Y})$ has zero probability. In log space, the exact MAP objective, becomes:

$$\max_{\theta} \quad \mathcal{L}(\theta) \quad + \log \mathbf{E}_{p_\theta(\mathbf{Y}|\mathbf{X})} \left[p_N(\mathbf{b}|\phi(\mathbf{X}, \mathbf{Y})) \right]. \tag{4.8}$$

Unfortunately with almost all noise models (including no noise), and box noise in particular, the second term in Equation 4.8 makes the optimization problem intractable. [2] Liang et al. [2009] use a variational approximation as well as a further approximation based on Jensen's inequality to reach the PR objective, which they ultimately optimize for their experiments. We also relate their framework to GE and CODL. If we approximate the last term in Equation 4.8 by moving the expectation inside the probability:

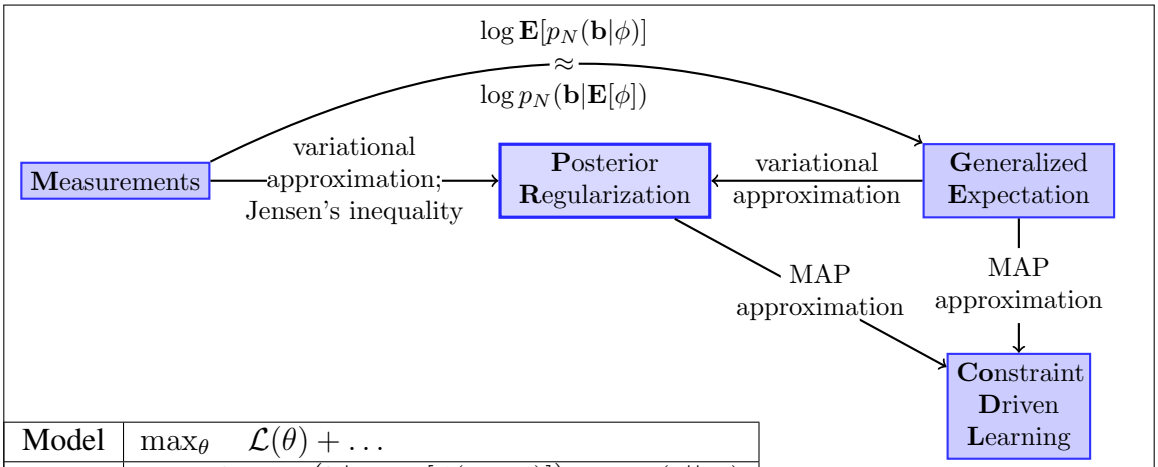
$$\mathbf{E}_{p_\theta(\mathbf{Y}|\mathbf{X})}\left[p_N(\mathbf{b} \mid \phi(\mathbf{X}, \mathbf{Y}))\right] \approx p_N\left(\mathbf{b} \mid \mathbf{E}_{p_\theta(\mathbf{Y}|\mathbf{X})}[\phi(\mathbf{X}, \mathbf{Y})]\right), \tag{4.9}$$

we end up with an objective equivalent to GE for appropriate noise models. In particular Gaussian noise corresponds to l_2^2 regularization in GE, since the log of a Gaussian is squared Euclidean distance (up to scaling). This approximation can be motivated by the case when $p_\theta(\mathbf{Y}|\mathbf{X})$ is concentrated on a particular labeling $\mathbf{Y}^*$: $p_\theta(\mathbf{Y}|\mathbf{X}) = \delta(\mathbf{Y}^*)$. In this special case the $\approx$ is an equality. This approximation is also used in Liang et al. [2009]. This provides an interpretation of GE as an approximation to the Bayesian framework proposed by Liang et al. [2009]:

$$\max_\theta \quad \mathcal{L}(\theta) \quad + \log p_N\left(\mathbf{b} \,\middle|\, \mathbf{E}_{p_\theta(\mathbf{Y}|\mathbf{X})}[\phi(\mathbf{X}, \mathbf{Y})]\right). \tag{4.10}$$

Note that the objective in Equation 4.10 is a reasonable objective in and of itself, essentially stating that the measured values $\mathbf{b}$ are not dependent on any particular instantiation of the hidden variables, but rather represent the integral over all their possible assignments. Liang et al. [2009] also use a variational approximation similar to the one of Bellare et al. [2009] so that the objective they optimize is exactly the PR objective, although their optimization algorithm is slightly different from the one presented in Chapter 2. Finally, if we restrict the set of allowable distributions further to be concentrated on a single labeling $\mathbf{Y}$, we recover the CODL algorithm. Figure 4.2 summarizes the relationships.

[2] For very special noise, such as noise that completely obscures the signal, we can compute the second term in Equation 4.8.



Model	$\max_\theta \quad \mathcal{L}(\theta) + \ldots$
PR	$\min_q \ \log p_N\left(\mathbf{b}|\mathbf{E}_{q(\mathbf{Y})}[\phi(\mathbf{X},\mathbf{Y})]\right) - \mathbf{KL}(q||p_\theta)$
M	$\log \mathbf{E}_{p_\theta(\mathbf{Y}|\mathbf{X})}\left[p_N(\mathbf{b}|\phi(\mathbf{X},\mathbf{Y}))\right]$
GE	$\log p_N\left(\mathbf{b}|\mathbf{E}_{p_\theta(\mathbf{Y}|\mathbf{X})}[\phi(\mathbf{X},\mathbf{Y})]\right)$
CODL	$\min_{\mathbf{Y}} \ \log p_N\left(\mathbf{b}|\phi(\mathbf{X},\mathbf{Y})\right) - \mathbf{KL}(\delta(\mathbf{Y})||p_\theta)$

Figure 4.2: A summary of the different models. We use $p_\theta(\mathbf{Y}|\mathbf{X})$ to denote the model probability, $q(\mathbf{Y})$ to denote a proposal distribution, and p_N for the noise model. The symbol $\delta(\mathbf{Y})$ denotes a distribution concentrated on $\mathbf{Y}$. The approximations are described in the text: M→GE near Equation 4.10, GE→PR near Equation 4.5, PR→CODL at the end of Section 4.3.

Chapter 5

Expressivity of the Language

Chapter 2 described the general form of the constraints that we are able to encode using the PR framework, and Part II is devoted to experiments with PR training of a variety of applications and models. Chapter 2 is general, but it is too abstract to achieve an intuition about what PR-style constraints are capable of expressing, while the chapters in Part II have too much details about the models, particular datasets and experimental settings to allow a reader to generalize an intuition about the expressivity of the linear constraints and penalties supported by PR. The goal of this chapter is to bridge this gap by providing a list of examples of the kinds of constraints that can and that cannot be encoded in PR for a few types of models and applications, as well as rules of thumb to help guide intuition. Since GE also uses the same kinds of constraints, it should be possible to implement the constraints we describe here efficiently in GE, and the constraints we cannot encode efficiently in PR should also not be efficiently encodable for GE.

For a more detailed treatment of the constraints we have used for specific applications, as well as more detailed descriptions of the models, experiments and result, see Chapters 6–10 in Part II.

The rest of this chapter is organized as follows: Section 5.1 describes some preliminaries and notational simplifications, and specifically defines a concept of decomposable constraints that we will use as a subset of efficiently computable constraints throughout the chapter.

Sections 5.2-5.4 describe a few constraints that can be encoded in a decomposable way

for a variety of models. For these constraints, we also provide variants that cannot be encoded in a decomposable way in order to give an intuition of the limitations of decomposable constraints. Section 5.2 focuses on labeled features which have been used extensively with GE; Section 5.3 describes agreement constraints such as the ones we use in Chapters 6 and 7; and Section 5.4 describes constraints that encode a sparsity structure, similar to the ones we use in Chapters 9 and 10.

Sections 5.5-5.6 focus on particular model structures rather than on particular constraints. Section 5.5 focuses on sequence models, while Section 5.6 focuses on dependency trees. For the two types of structures, they describe some example constraints that are decomposable and some that are not decomposable with respect to those model structures. In all cases, we only present a few examples rather than a complete characterization with the hope of giving the reader an intuition of what kinds of constraints might be decomposable for their application.

5.1 Preliminaries

Chapter 2 defines a constraint set with respect to expectations of constraint features. While Equation 2.2 is very general, it is worth considering simplifications which are more concise and hence easier to think about. Figure 5.1 shows a few simplifications of Equation 2.2. The constraint sets in Figure 5.1 are written in terms of distributions $q(\mathbf{Y})$ over the entire corpus. This allows the formulation to be more general, but might be a disservice to intuition. Often, we would like to make statements about distributions within each instance. The running example in Chapter 2 is one such case. In these cases, we can create corpus constraints by simply stacking vectors of instance constraints on top of each other to create a high-dimensional vector of constraint features and feature expectations. This means that:

Equality	$\mathcal{Q} = \{q(\mathbf{Y}) : \mathbf{E}_q[\phi(\mathbf{X}, \mathbf{Y})] = \mathbf{b}\}$.
Inequalities	$\mathcal{Q} = \{q(\mathbf{Y}) : \mathbf{E}_q[\phi(\mathbf{X}, \mathbf{Y})] \leq \mathbf{b}\}$.
Equality with slack	$\mathcal{Q} = \{q(\mathbf{Y}) : \|\|\mathbf{E}_q[\phi(\mathbf{X}, \mathbf{Y})] - \mathbf{b}\|\|_\beta \leq \epsilon\}$.
Inequalities with slack	$\mathcal{Q} = \{q(\mathbf{Y}) : \exists\, \xi,\ \mathbf{E}_q[\phi(\mathbf{X}, \mathbf{Y})] - \mathbf{b} \leq \xi;\ \|\|\xi\|\|_\beta \leq \epsilon\}$.

Figure 5.1: Summary of the simplified constraints. The last row is the most general (and hence most powerful) but also hardest to read. Many interesting constraints can be encoded with simple equality.

$$\phi(\mathbf{X}, \mathbf{Y}) = \begin{bmatrix} \phi_1(\mathbf{y}_1, \mathbf{x}_1) \\ \vdots \\ \phi_i(\mathbf{y}_i, \mathbf{x}_i) \\ \vdots \\ \phi_n(\mathbf{y}_n, \mathbf{x}_n) \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_{\mathbf{x}_1} \\ \vdots \\ \mathbf{b}_{\mathbf{x}_i} \\ \vdots \\ \mathbf{b}_{\mathbf{x}_n} \end{bmatrix} \tag{5.1}$$

where $n = |\mathbf{X}|$ is the number of instances in our corpus. If the constraints have this special form and the model factors as a product distribution over instances, then we can perform the KL-projection for each instance independently of all the other instances and the resulting solution to the projection will be just the product distribution of the instance-specific ones:

$$q^*(\mathbf{Y}) = \prod_i q^*_{\mathbf{x}_i}(\mathbf{y}_i), \tag{5.2}$$

where $\mathbf{y}_i$ is the portion of $\mathbf{Y}$ corresponding to the i^{th} instance in the corpus, $q^*(\mathbf{Y})$ is the solution to the minimization of $\mathbf{KL}(q(\mathbf{Y})||p_\theta(\mathbf{Y}|\mathbf{X}))$ subject to the constraints and $q^*_{\mathbf{x}_i}(\mathbf{y}_i)$ is the solution of the minimization $\mathbf{KL}(q(\mathbf{y})||p_\theta(\mathbf{y}|\mathbf{x}_i))$. In cases where it is natural to define constraints on an instance-by-instance basis, we will do so. In order to simplify notation, we will sometimes drop the i and $\mathbf{x}_i$ subscripts when it is clear what they should be.

Note that we use the norm notation throughout the thesis even though any convex function whose dual can be efficiently evaluated and optimized will also work. In this chapter we mostly focus on the constraint features $\phi(\mathbf{X}, \mathbf{Y})$. If a constraint set is efficiently encodable with respect to a model $p_\theta(\mathbf{Y}|\mathbf{X})$ then by Proposition 2.2 we can use the same inference algorithms to compute feature expectations for $q(\mathbf{Y})$ as we do for $p_\theta(\mathbf{Y}|\mathbf{X})$. The following subsection makes this more precise.

5.1.1 Decomposable constraints

While the most general form of Equation 2.2 is very powerful, having general constraint features $\phi(\mathbf{Y})$ might not always result in a tractable optimization problem. In fact, as we will see it will sometimes be NP-hard or ♯P-hard to compute the optimal $q^*(\mathbf{Y})$ subject to $\mathbf{E}_q[\phi(\mathbf{X}, \mathbf{Y})] = \mathbf{b}$, or the optimal $q^*(\mathbf{Y})$ will not be representable efficiently. Even when the exact projection is not tractable it might be possible to approximate it, for example by using Monte-Carlo samples as was done by Bellare et al. [2009].

Rather than attempting to characterize the full range of possibilities (and failing), in this chapter we will focus on using Proposition 2.2 to guarantee that it is possible to use the same machinery we have for computing sufficient statistics in $p_\theta(\mathbf{Y}|\mathbf{X})$ to compute them in $q(\mathbf{Y})$.

Definition 5.1 (decomposable constraint)**.** *Suppose that our model's conditional distribution over labels $p_\theta(\mathbf{Y}|\mathbf{X})$ can be represented as a product over some clique potentials $p_\theta(\mathbf{Y}|\mathbf{X}) \propto \prod_{c \in \mathcal{C}} \psi_c(\mathbf{Y}_c|\mathbf{X})$. Then we say that a constraint set $\mathcal{Q}$ is* decomposable *with respect to the model if it can be encoded as Equation 2.2 (Constraint set with slack) where the constraint features decompose as:*

$$\phi(\mathbf{X}, \mathbf{Y}) = \sum_c \phi_c(\mathbf{X}, \mathbf{Y}_c). \tag{5.3}$$

Note that if the model distribution $p_\theta(\mathbf{Y}|\mathbf{X})$ decomposes as a product over instances, an instance-specific version of Definition 5.1 is strictly stronger than the corpus-wide version in Definition 5.1. For simplicity, it will sometimes be easier to use an instance specific notation: if $p_\theta(\mathbf{y}|\mathbf{x}) \propto \prod_{c \in \mathcal{C}} \psi_c(\mathbf{y}_c|\mathbf{x})$ and $\phi(\mathbf{x}, \mathbf{y}) = \sum_c \phi_c(\mathbf{x}, \mathbf{y}_c)$, we can define $\phi(\mathbf{X}, \mathbf{Y})$ as in Equation 5.2 and $\mathcal{Q}$ will be decomposable with respect to $p_\theta(\mathbf{Y}|\mathbf{X})$.

Note also that for efficiency of computing the projection in its dual formulation, it is important for us to be able to efficiently project onto or optimize with respect to the dual norm $||\cdot||_{\beta^*}$. We do not consider very complicated norms in this chapter, so a precise definition of this will not be necessary for the purpose of developing an intuition.

5.2 Feature labeling

The form of Equation 5.3 suggests one very versatile type of constraint, sometimes called *feature labeling* [Druck et al., 2008]. This is most easily viewed from the point of view of a log-linear model, such as a maximum entropy model [Jaynes, 1957, Good, 1963] or conditional random field [Lafferty et al., 2001]. Recall that the form of the model for a conditional random field (CRF) is given by:

$$p(\mathbf{y}) \propto \exp(\mathbf{f}(\mathbf{x}, \mathbf{y}) \cdot \theta) \tag{5.4}$$

where in order for the normalizer of $p(\mathbf{y})$ to be efficiently computable, $\mathbf{f}(\mathbf{x}, \mathbf{y})$ must decompose as a sum across the cliques of the model, and the model must have low treewidth. Consequently, any feature function that we would be able to use for a CRF or maximum entropy model would be a decomposable constraint feature $\phi(\mathbf{x}, \mathbf{y})$.

For example, Druck et al. [2009] train a dependency parser where they ask a user to label some features of the form "given that a sentence has a verb and a noun, what is the probability that the verb dominates the noun?" The user then states a constraint in the form of an approximate probability for that event. In earlier work Druck et al. [2008] give the following example of a feature labeling constraint. Suppose we are interested in classifying documents into the categories "baseball" vs. "hockey." The word "puck" appearing in the document should give us a fairly strong signal that the correct label is "hockey." We can encode this knowledge as the constraint that for the collection of articles where the word "puck" appears the probability of a label of "hockey" should be at least α, where α is a confidence value – e.g. 95%. Specifically to encode this using our notation we would have:

$$\phi(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{|\mathbf{X}|} \begin{cases} 1 & \text{if article } \mathbf{x}_i \text{ contains the word "puck" and } \mathbf{y}_i = \text{"hockey"} \\ 0 & \text{otherwise,} \end{cases} \tag{5.5}$$

and require that the expectation of this feature to be at least an α fraction of the articles that contain the word "puck":

$$\mathbf{E}_q\left[\phi(\mathbf{X}, \mathbf{Y})\right] \geq \sum_{i=1}^{|\mathbf{X}|} \begin{cases} 1 & \text{if article } \mathbf{x}_i \text{ contains the word "puck"} \\ 0 & \text{otherwise.} \end{cases} \tag{5.6}$$

It is worth noting that even though there might be a large number of articles, we have chosen to encode the information as a single constraint. The alternative would have been to include a constraint for each article that contains the word "puck" and require that the expectation of each of these constraints is at least α. However, this would have required that each article has a label of "hockey" with probability at least α, rather than requiring that an α fraction of the articles are labeled as "hockey" in expectation. In the multiple constraint case, we have to prefer "hockey" over "baseball" for each article, while in the single constraint case, we can prefer "baseball" over "hockey" in a few articles as long as there are a sufficient number where we strongly prefer "hockey" over "baseball." For this particular piece of prior knowledge, a single corpus-wide constraint is more appropriate.

These kinds of feature-labeling constraints have been used extensively by a number of researchers [Liang et al., 2009, Druck et al., 2008, 2009, Mann and McCallum, 2007, Bellare et al., 2009, Mann and McCallum, 2008]. The difficulty in creating them is that in choosing α for a number of features might be difficult. In the dependency parsing verb-noun example above it might not be immediately clear what the probability that a verb dominates a noun should be. If there is only one noun and one verb, then probably it should be close to one. In a sentence with only one noun and several verbs, it is not clear which of the noun-verb links should be active. One method for gathering the constraint feature expectations is in a semi-supervised setting where in addition to unlabeled data, we are given a labeled corpus. Liang et al. [2009] generate feature expectations in this way and report significant improvements in performance. Quadrianto et al. [2009] use a similar type of constraint, although in a slightly different framework and also report state of the art results.

An alternative source of feature labeling constraints would be a rule-based system for solving the problem. For example, suppose that we have available a rule-based part-of-speech tagger, that we are confident has an accuracy of at least 95%. Additionally, we might have a small labeled corpus. If we want to combine these two resources, we could use the PR framework to train a tagger on the labeled data, but use the constraint that on unlabeled data, our learned tagger should agree with the rule-based tagger at on at least 95% of the part-of-speech tags, in expectation. We could use other sources of noisy la-

bels: Chapter 8 describes our experiments with training both discriminative and generative dependency parsers using parallel text and a foreign language parser as a source of noisy labels. Other possibilities include labels obtained from untrained annotators, or game-based data collection methods. It is also possible to get similar feature labels from related tasks. For example, if we want to train a part-of-speech tagger, but have a corpus annotated with named entities, we could add a constraint that the tokens corresponding to named entities should have proper noun, noun or adjective tags 95% of the time. If instead we want to induce both a part-of-speech tagger and a named entity recognizer, we could include agreement constraints as described in the next section.

5.3 Agreement and Disagreement

The second category of constraints we will discuss are what we call agreement constraints, and related disagreement constraints. Chapter 7 describes an application of agreement constraints where we have two views of the data, and we would like the predictions of two models to agree on the label probabilities for each label. Chapter 6 presents another type of agreement constraint which joins alignment models that have different structure (one models a source language as observations while another as hidden states), but requires the two models to agree on the probability that a particular alignment link exists. In this section we compare a few agreement constraints that can be encoded using decomposable constraint features to some similar constraints that cannot. Table 5.1 summarizes the conclusions of this section.

5.3.1 Two Views

Chapter 7 describes a setting where we have two models over the same output variables and we would like to prefer that the two models agree on the probabilities of different labels. Chapter 7 describes the closed form solution, as well as the relation to the Bhattacharyya distance as a regularizer and also shows how the agreement between two views can be enforced in more complicated situations. For simplicity, in this section we will focus on a binary classification problem. Suppose for a particular instance, we have two models of

Name	Setting	What we can say	What we can't say
Two views	We have two models over the same set of labels, as in Chapter 7.	The probability of each label should be similar for the two models.	The models should produce the same label with high probability. (See text).
Graph structure	Clustering or document classification where instances are nodes in a graph.	Prefer to label instances in the same cluster/class if they are connected by an edge. More weakly prefer to label them differently.	Prefer to label instances in different clusters/classes so that the most likely labels disagree.
Symmetry	As in Chapter 6: Two word alignment models differ according to which language they treat as hidden states and which language they treat as observations.	Two different models should agree on the probability of alignment links.	Two different models should agree on the probability of entire alignments.

Table 5.1: Summary of some agreement constraints that are efficiently encodable in the PR framework along with some similar agreement constraints that cannot be encoded efficiently.

what the output should be: $p_1(y_1)$ and $p_2(y_2)$. The constraint described in Chapter 7 is equivalent to: Find a joint distribution $q(y_1, y_2)$ such that $\mathbf{KL}(q||p_1p_2)$ is small, subject to the constraint that $q(y, \cdot) = q(\cdot, y)$ for all y, where $q(y, \cdot)$ represents the marginal distribution of y in the first position $q(y, \cdot) = \sum_{y'} q(y, y')$ and analogously for $q(\cdot, y)$. More concisely:

$$\min_q \ \mathbf{KL}(q(y_1, y_2)||p_1(y_1)p_2(y_2)) \quad \text{s.t.} \ \sum_{y'} q(y, y') = \sum_{y'} q(y', y) \ \ \forall y, \tag{5.7}$$

where $p_1(y_1)p_2(y_2)$ is a product distribution, and the y, y_1 and y_2 range over the same possible values. There are several equivalent ways to write this problem, and a closed form solution exists. See Chapter 7 for some details. It is also easy to solve the problem with slack:

$$\min_q \ \mathbf{KL}(q(y_1, y_2)||p_1(y_1)p_2(y_2)) \quad \text{s.t.} \ \left| \sum_{y'} q(y, y') - q(y', y) \right| \leq \epsilon \ \ \forall y. \tag{5.8}$$

Note that here, the form of the model p_θ is a product distribution over the two labels $p_\theta(y_1, y_2) = p_1(y_1)p_2(y_2)$. Consequently, we need to be able to write the constraint features in a way that they factor as a sum over the two labels. Luckily, the constraint can be written as $|\mathbf{E}_q[\phi_1(y_1) - \phi_2(y_2)]| \leq \epsilon$, where $\phi_1(y_1)$ returns a vector with 1 at position y_1 and zeros everywhere else, and $\phi_2(y_2)$ returns a vector with a 1 at position y_2 and zeros elsewhere. Chapter 7 shows that this can be extended to structured models (when $\epsilon = 0$), so that even if there are exponentially many possible labels y, the optimization problem in Equation 5.7 still has a closed form solution. In Section 5.3.3, we will see that if the models do not share the same structure computing the projection can be hard. A formulation related to that in Equation 5.8, but not encodable with decomposable constraints is given by:

$$\min_q \ \mathbf{KL}(q(y_1, y_2)||p_1(y_1)p_2(y_2)) \quad \text{s.t.} \sum_{y_1} \sum_{y_2 \neq y_1} q(y_1, y_2) \leq \epsilon. \tag{5.9}$$

Stating the two constraints in English might make the distinction clearer: The constraint in Equation 5.8 states "the distribution of labels in the first and second position should agree." While, the constraint in Equation 5.9 states "if we choose labels from q, the labels in the first and second position should agree with high probability." The constraint in Equation 5.9 cannot be written as a sum over constraint features that each only consider one of the labels. Intuitively, this is because we need to specifically consider whether $y_1 = y_2$ in order to evaluate whether a violation has occurred. Consequently, $q(y_1, y_2)$ will not factor as a product distribution. If we wanted to add an additional constraint that $q(y_1, y_2) = q_1(y_1)q_2(y_2)$, we would have a problem which is not convex (in q_1 and q_2) as shown in Figure 5.2.

Figure 5.2 illustrates the following setting: y ranges over two possible outcomes, so p_1 and p_2 each have one free parameter (a "heads" probability). If we require $q(y_1, y_2) = q_1(y_1)q_2(y_2)$, then q_1 and q_2 also each have one free parameter, and we can plot settings of that parameter for which the constraints in Equations 5.8 and 5.9 are satisfied. With the additional requirement that $q(y_1, y_2) = q_1(y_1)q_2(y_2)$ the constraint set analogous to Equation 5.9 is not convex (or even connected).

Note that the constraint set in Equation 5.9 is linear in $q(y_1, y_2)$, since there is no requirement for $q(y_1, y_2)$ to factorize. For unstructured problems it would be efficient to

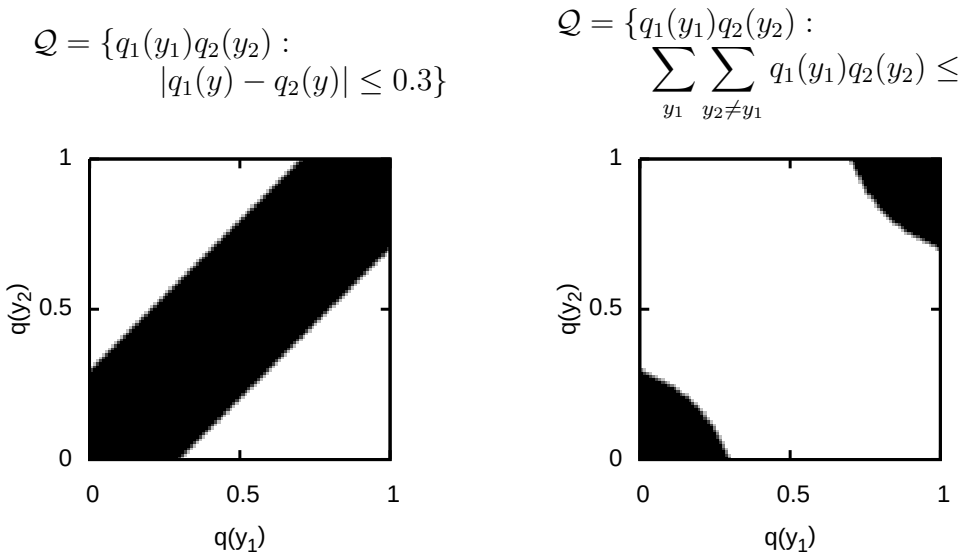


Figure 5.2: Illustration of two possible constraints for the optimization problem $\min_{q_1,q_2} \mathbf{KL}(q_1(y_1)q_2(y_2)||p_1(y_1)p_2(y_2))$ s. t. $q \in \mathcal{Q}$. Left: constraints similar to Equation 5.8; Right: constraints similar to Equation 5.9.

solve 5.9, since we would need to enumerate only quadratically many possibilities. However, the constraint set does not fit Definition 5.1, and for structured models the optimization in Equation 5.9 cannot be solved by enumeration.

5.3.2 Graph Structure

Possibly the simplest kind of agreement constraint is motivated by graph-based semi-supervised learning [Chapelle et al., 2006, and references therein]. In addition to any labeled and unlabeled instances that we wish to classify, we receive a graph whose vertices correspond to the instances in the labeled and unlabeled corpora. Commonly, we would like to enforce that the labels our method assigns to two instances connected by an edge are not very different. Here, $p_\theta(\mathbf{Y})$ is a model that defines a probability distribution over labelings of the entire corpus. However, the distribution decomposes as a product over instances $p_\theta(\mathbf{Y}) = \prod_i p_\theta(\mathbf{y}_i)$. Let E be the set of edges in the graph. Then we could encode the desired preference as:

$$\min_q \mathbf{KL}(q(\mathbf{Y})||p(\mathbf{Y})) \quad \text{s. t.} \quad |\mathbf{E}_q[\phi(\mathbf{y}_i) - \phi(\mathbf{y}_j)]| \leq \epsilon \ \forall (i,j) \in E, \tag{5.10}$$

where $\phi(\mathbf{y})$ is the vector of indicators

$$\phi(\mathbf{y}) = \begin{bmatrix} \delta(\mathbf{y}=0) \\ \delta(\mathbf{y}=1) \\ \vdots \\ \delta(\mathbf{y}=n) \end{bmatrix} \quad \text{with} \quad \delta(x) = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{otherwise} \end{cases}. \tag{5.11}$$

Note that if we have $\epsilon = 0$, the problem is reduced to choosing one distribution over labels for every connected component of the graph.

It is not generally possible to encode disagreement in the same form as Equation 5.10. In particular the problem:

$$\min_q \mathbf{KL}(q(\mathbf{Y})||p(\mathbf{Y})) \quad \text{s.t.} \quad |\mathbf{E}_q[\phi_i(\mathbf{Y}) - \phi_j(\mathbf{Y})]| \geq \epsilon \ \ \forall (i,j) \in E \tag{5.12}$$

does not have a convex constraint set, and is NP-hard to solve in general, by a reduction from three-coloring. To see this, set $\epsilon = 1$, suppose that each node chooses from one of three labels (representing the three colors) and let $p(\mathbf{Y})$ be the uniform distribution. The constraint in Equation 5.12 requires any two nodes that are connected by an edge to be colored differently, and so any q that satisfies the constraints implies the existence of a three-coloring of the graph. In the special case where there are only two labels, a strong disagreement constraint can be encoded as:

$$\min_q \mathbf{KL}(q(\mathbf{Y})||p(\mathbf{Y})) \quad \text{s.t.} \quad \begin{array}{l} |\mathbf{E}_q[\phi_{i,0}(\mathbf{Y}) - \phi_{j,1}(\mathbf{Y})]| \leq \epsilon \\ |\mathbf{E}_q[\phi_{i,1}(\mathbf{Y}) - \phi_{j,0}(\mathbf{Y})]| \leq \epsilon \end{array} \forall (i,j) \in E. \tag{5.13}$$

Where $\phi_{i,0}(\mathbf{Y})$ takes on value 1 if an only if $\mathbf{Y}_i = 0$ and respectively for $\phi_{i,1}(\mathbf{Y})$, $\phi_{j,0}(\mathbf{Y})$ and $\phi_{j,1}(\mathbf{Y})$. Equation 5.13 uses the fact that we know that if the label of node i is 1 and it must disagree with node j, then the label of node j must be 0.

However, it is possible to encode a weaker notion of disagreement. Specifically, we can say that the sum of the probabilities of labels on two instances should be at most one.

$$\min_q \mathbf{KL}(q(\mathbf{Y})||p(\mathbf{Y})) \quad \text{s.t.} \quad \mathbf{E}_q[\phi(\mathbf{y}_i) + \phi(\mathbf{y}_j)] \leq 1 + \epsilon \forall (i,j) \in E. \tag{5.14}$$

Note that this is a particularly weak notion of disagreement. If there are many possible labels, it is likely that $p_\theta(\mathbf{y}_i) \leq 0.5$ for all possible labels, and the constraints will be

automatically satisfied. For example with $\epsilon = 0$, Equation 5.10 requires that two nodes joined by an edge have the same most likely label according to q, while Equation 5.14 does not require that two nodes joined by an edge have different most likely labels.

5.3.3 Symmetry

Section 6.3 describes symmetry constraints for word alignment. In this case, we have a parallel sentence pair and two hidden Markov models, with a different number of observations and hidden states. For example, suppose we have a Nepali sentence with n words and a Malay sentence with m words. We will have one model, called Nepali→Malay, whose hidden states correspond to the Nepali words and whose observations correspond to the Malay words. We will have a second model called Malay→Nepali, whose hidden states correspond to the Malay words and whose observations correspond to the Nepali words. Hence the Nepali→Malay model will have n hidden states with m observations, and vice-versa for the Malay→Nepali model. The interpretation of the hidden state assignment is that if the hidden state corresponding to word i in one language generates the observation corresponding to word j in the other language, then these two words are translations of each other in this context. Chapter 6 describes details about the model parameterization, however we ignore these details to focus on expressibility of constraints in this setting using the PR framework.

Definition 5.2 (Alignment Link)**.** *An* alignment link *in this context is a pair* (i, j) *with* $i \in \{1 \dots n\}$ *and* $j \in \{1 \dots m\}$ *corresponding to the statement that the* i^{th} *Nepali word is the translation of the* j^{th} *Malay word in the sentence pair.*

Definition 5.3 (Alignment)**.** *An* alignment *in this context is a complete set of alignment links, specifying which Malay words translate to which Nepali words. This can be represented as an* $n \times m$ *binary matrix with entry* (i, j) *being 1 iff the* (i, j) *alignment link is part of the alignment.*

Both the Malay→Nepali and the Nepali→Malay models induce a distribution over alignments. Given such a distribution, we can compute a probability that an alignment link

(i, j) is present in an alignment generated by this distribution. For a hidden Markov alignment model, the probability of an alignment link corresponds to the posterior probability of a particular state at a particular position in the HMM. The symmetry constraints described in Section 6.3 encode the statement: for all i, j, the Malay→Nepali and the Nepali→Malay models should give the same probability that the alignment link (i, j) is present. This constraint can be encoded by creating features $\overrightarrow{\phi}_{i,j}(\mathbf{y})$ that correspond to the (i, j) alignment link being present according to the Nepali→Malay model and features $\overleftarrow{\phi}_{i,j}(\mathbf{y})$ that correspond to the (i, j) alignment link being present in according to the Malay→Nepali model. Then the constraint states $\overrightarrow{\phi}_{i,j}(\mathbf{y}) - \overleftarrow{\phi}_{i,j}(\mathbf{y}) = 0$, so the constraint set is decomposable with respect to a product distribution over the two directions.

An alternative statement that we might want to encode is: for all possible alignments, the probability of the alignment according to the Malay→Nepali model should be equal to its probability under the Nepali→Malay model. Perhaps surprisingly, it is ♯P-hard to normalize a distribution with this constraint. This might be surprising because in Chapter 7 we observe that a similar statement for two-view learning can be computed in closed form. However, because the structure of the Malay→Nepali model is different from the structure of the Nepali→Malay model, the problem becomes hard.

The proof sketch of the ♯P-hardness is by a simple reduction from the problem of computing the permanent, or equivalently of counting the number of perfect matchings in a bipartite graph. Given a bipartite graph with n vertices in each part, construct a Malay→Nepali model that has uniform transition probabilities and non-zero emission probability for hidden state i to generate observed state j if and only if the bipartite graph has a link from node i in the left part to node j in the right part. Similarly, construct a Nepali→Malay model with uniform transition probabilities and non-zero emission probability for hidden state j to generate observed state i if and only if the bipartite graph has a link from node i in the left part to node j in the right part. For both models, let the non-zero probabilities be uniform. Now an alignment can be interpreted as a subset of the edges in the original bipartite graph, and the Malay→Nepali model gives a uniform distribution over subsets of the edges that use the right part exactly once, while the Nepali→Malay model gives a uniform distribution over subsets of the edges that use the left part exactly

once. The intersection of the non-zero alignments of the two models encodes exactly the set of perfect matchings, and if we could compute the uniform distribution over these, we would be able to count the number of perfect matchings. However, counting the number of perfect matchings in a bipartite graph is ♯P-hard [Valiant, 1979].

5.4 Sparsity Structure

Chapters 9 and 10 describe the application of PR for preferring models that display a particular form of sparsity structure in the posteriors. This section describes these sparsity penalties and introduces a few more possible applications of the idea. In the simplest setting, suppose that we have an unstructured model, and a corpus of instances: for example, we might be interested in classifying scientific articles into a number of topics. The prior knowledge that we want to encode is that some sets of the instances should collectively only cover a small fraction of the topics. For example, we might know the names of the authors of each article. We know that each author only publishes on a small subset of the different possible topics. Let A_i be the set of authors of document i, and let a range over authors. Assume that we know A_i for each document i. Fix a labeling $\mathbf{Y}$. We say that an author a publishes in a topic y if there is some document published by an author with topic y. Call this event

$$\phi_{ay}(\mathbf{Y}) = \begin{cases} 1 & \text{if } \exists i \text{ s.t. } a \in A_i \text{ and } \mathbf{Y}_i = y \\ 0 & \text{otherwise} \end{cases}. \tag{5.15}$$

A reasonable constraint we might want to enforce is

$$\sum_{ay} \mathbf{E}_q[\phi_{ay}(\mathbf{Y})] \leq \epsilon. \tag{5.16}$$

Note that while this constraint penalizes all author-topic pairs, it does not penalize every document-topic pair. That is, if an author has published in a particular topic, we will not pay an additional penalty for other publications by the same author in the same topic; if an author has never published in a topic then we pay a penalty for the new topic-author pair. The model $p_\theta(\mathbf{Y})$ in this case decomposes as a product over labels of individual

documents, but unfortunately the constraint in Equation 5.16 does not decompose as a sum over different instances. Instead, we can define slightly more decomposed features

$$\phi_{ayi}(\mathbf{Y}) = \begin{cases} 1 & \text{if } a \in A_i \text{ and } \mathbf{Y}_i = y \\ 0 & \text{otherwise} \end{cases}, \tag{5.17}$$

which are a refinement in the sense that ϕ_{ay} is 1 if there is some i with ϕ_{ayi} being 1: $\phi_{ay}(\mathbf{Y}) = 1$ iff $\exists i\ \phi_{ayi}(\mathbf{Y}) = 1$. Note that for any fixed labeling $\mathbf{Y}$,

$$\max_i \phi_{ayi}(\mathbf{Y}) = \phi_{ay}(\mathbf{Y}) \tag{5.18}$$

and we approximate the expectation of ϕ_{ay} with the max of the expectations of ϕ_{ayi}. The relaxed constraints are:

$$\sum_{ay} \max_i \mathbf{E}_q[\phi_{ayi}(\mathbf{Y})] \leq \epsilon. \tag{5.19}$$

A second potential criticism is that we are summing over all labels rather than counting the non-zero entries. This could be encoded as

$$\sum_{ay} \delta\left(\max_i \mathbf{E}_q[\phi_{ayi}(\mathbf{Y})] > 0\right) \leq \epsilon, \tag{5.20}$$

where $\delta(\cdot)$ takes value 1 if its argument is true and 0 otherwise. Unfortunately, the formulation in Equation 5.20 is not convex in q. To see this note that if we have just one instance and one author, constraint requires at most ϵ labels to have non-zero probability according to q. If $\epsilon = 1$, then $q(\mathbf{y} = 1) = 1$ is inside $\mathcal{Q}$, and $q(\mathbf{y} = 2) = 1$ is inside $\mathcal{Q}$, but their convex combination $q(\mathbf{y} = 1) = q(\mathbf{y} = 2) = 0.5$ is not.

The rest of this section lists some possible applications of the sparsity constraints.

1. Clustering: suppose we have a set of images of actor's faces from many episodes of many different TV programs. The goal is to cluster the faces into a number of actors. For concreteness, suppose we know in advance the number of clusters and we will use Gaussian clustering in the space of eigenfaces [Sirovich and Kirby, 1987]. We might want to say: each TV show should correspond to a small number of actors. We can do this by introducing a sparsity constraint as in Equation 5.19.

2. Topic modeling: Suppose we have a large number of blogs, and we want to learn a model for the topics of each post, for example using a latent Dirichlet allocation topic model [Blei et al., 2003]. We can use PR with sparsity constraints to require that each blog only uses a small number of subjects throughout all its posts.

3. Sequence models: Chapter 9 describes our experiments for trying to induce part-of-speech tags using a hidden Markov model, and a PR penalty term similar to the one in Equation 5.19 This leads to a smaller number of possible tags for each word, and improves the usefulness of the tags.

4. Dependency trees: Chapter 10 describes our experiments on grammar induction, where we try to train a dependency model and use a PR penalty term to make the model prefer models that have a fewer number of possible dependency edge types.

5.5 Sequence Models

This section and the next one focus on particular classes of structured models and describe what types of information can be encoded with decomposable constraints for these models and what prior knowledge either cannot be encoded, or there is no clear way to encode them. This section focuses on models where the hidden variables have a sequential structure, while the following section focuses on cases where the hidden variables are dependency trees.

Formally, a sequence model is one where the output variables are composed of a sequence of labels $\mathbf{y} = (y_1 \dots y_n)$. For simplicity of exposition, we will assume that n is a fixed length and where the conditional model distribution can be encoded as a first-order Markov model:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \prod_{i=0}^{n} \psi_i(y_i, y_{i+1}, \mathbf{x}), \tag{5.21}$$

where the factor ψ_i may be a function of the entire observed variables $\mathbf{x}$ but can only depend on two consecutive positions of the output $\mathbf{y}$. Higher-order Markov dependencies are also possible, as are semi-Markov models, and for some applications the additional modeling power is very important, however the observations about decomposability of

Name	Application	Constraint description
Labeled Features	Any	Given a particular rule that provides a partial labeling, this rule should be correct at least α fraction of the time over the entire corpus. Multiple rules can be incorporated by enforcing multiple constraints.
Expected Count	POS Tagging	There should be at least as many verbs as sentences in the corpus. Similarly for nouns or other labels in other applications.
Transition	POS Tagging	Determiners should be followed by a noun or adjective 90% of the time. Adjectives should be followed by nouns or other adjectives 90% of the time.
Composition	CCG supertagging	Consecutive pairs of CCG supertags should be able to combine with eachother 90% of the time.

Table 5.2: Summary of simple constraints that are possible for sequences. In all cases, $\phi_{i,j}(\mathbf{y})$ takes on the value 1 if the tree $\mathbf{y}$ contains an edge $i \rightarrow j$ and 0 otherwise. Consequently, $\mathbf{E}_q[\phi_{i,j}(\mathbf{y})]$ is the posterior probability that an edge is present in a tree drawn from q.

constraints carries over to higher order Markov models, while semi-Markov models allow a few additional types of constraints.

Sequence models have been used for a wide variety of tasks in natural language processing, signal processing and bioinformatics. Canonical natural language applications include part-of-speech tagging, shallow parsing and information extraction [DeRose, 1988, Church, 1988, Merialdo, 1994, Lafferty et al., 2001, Sang and Buchholz, 2000b, Manning and Schütze, 2000]. Signal processing applications include speech recognition, handwriting recognition, and optical character recognition [Juang and Rabiner, 1991, Taskar et al., 2004]. Bioinformatics applications include gene finding [Burge and Karlin, 1997, Bernal et al., 2007].

Examples of models whose conditional distribution follows the form of Equation 5.21 include first-order hidden Markov models [Rabiner, 1989], conditional random fields [Lafferty et al., 2001] and Markov random fields [Kindermann et al., 1980]. The form of Equa-

tion 5.21 requires that decomposable constraints should have constraint features that decompose as a sum of features that depend on the label of a single position y_i or on a pair of consecutive positions (y_i, y_{i+1}).

Table 5.2 summarizes some constraints that can be encoded in a decomposable way with respect to a first-order Markov model. For simplicity, let us focus on a part-of-speech tagging problem, so that the label at any position, y_i, is a part-of-speech tag. In this setting there are a number of constraints that we can encode.

As a first example, suppose we know that noun phrases tend to be composed of an optional determiner, followed by an optional sequence of adjectives, followed by a sequence of nouns. Suppose we know that the word "the" is almost always a determiner. This is an example of a feature-labeling constraint as described in Section 5.2. It is decomposable with respect to the model defined in Equation 5.21, since we can encode it as $\mathbf{E}_q[\phi(\mathbf{Y})] \geq \epsilon c$, where c is the number of nouns in the corpus, $\phi(\mathbf{Y})$ counts the number of occurrences of "the" labeled as a determiner and ϵ controls the confidence. Since $\phi(\mathbf{Y})$ is a sum of indicator functions for the determiner, one at each occurrence of "the", the constraint is decomposable (Definition 5.1).

Suppose we want to ensure that in every sentence, a noun appears at some point after the appearance of the word "the". We can encode this by defining a constraint feature for every occurrence of the word "the" in the corpus that counts the number of nouns after that occurrence of "the". Define the features:

$$\phi_i(\mathbf{y}) = \begin{cases} 1 & \text{if } y_i = \text{“noun”} \\ 0 & \text{otherwise.} \end{cases} \tag{5.22}$$

Then for all positions i such that the word at position i is "the", we can require that the expected value of the sum of all features ϕ_j with $j > i$ is at least 1. The optimization problem then becomes:

$$\min_q \mathbf{KL}(q(\mathbf{y})||p(\mathbf{y})) \quad \text{s.t.} \quad \mathbf{E}_q\left[\sum_{j>i} \phi_j(\mathbf{y})\right] \geq 1 \; \forall i : x_i = \text{“the”}. \tag{5.23}$$

Note that this constraint is decomposable according to Definition 5.1, since we are taking the expectation of a sum of $\phi_j(\mathbf{y})$, and each of the constraint features $\phi_j(\mathbf{y})$ only consider a single token and its label.

A related constraint that is not decomposable and cannot be encoded efficiently is the following: With high probability there should be at least one noun somewhere in the labeling after the occurrence of "the". Writing this optimization problem we have:

$$\min_q \mathbf{KL}(q(\mathbf{y})||p(\mathbf{y})) \quad \text{s.t.} \quad \mathbf{E}_q\left[\phi'_i(\mathbf{y})\right] \geq 1 \; \forall i : x_i = \text{“the”}. \tag{5.24}$$

where

$$\phi'_i(\mathbf{y}) = \begin{cases} 1 & \text{if } \exists j > i \text{ with } y_j = \text{“noun”} \\ 0 & \text{otherwise.} \end{cases}$$

The problem with the latter constraint of Equation 5.24 is that in order to evaluate whether the labeling contains a noun at some point after "the", we need to consider at the same time all the parts-of-speech of the words following "the". Because we essentially need a logical "or" of the tags of a long sequence of words, we cannot express ϕ'_i as a sum of features that only consider the label of one word or pairs of words, and the constraint is not decomposable according to Definition 5.1.

If we believe the determiner-adjective-noun pattern, we could add some additional constraints. For example, we might want to encourage determiners to be followed at some later point by nouns. We can do this by defining an extra constraint feature ϕ^{D} similar to the ones in Equation 5.22:

$$\phi^{\mathrm{D}}_i(\mathbf{y}) = \begin{cases} 1 & \text{if } y_i = \text{“determiner”} \\ 0 & \text{otherwise} \end{cases}. \tag{5.25}$$

We would then optimize the following problem:

$$\min_q \mathbf{KL}(q(\mathbf{y})||p(\mathbf{y})) \quad \text{s.t.} \quad \mathbf{E}_q\left[\phi^{\mathrm{D}}_i(\mathbf{y}) - \sum_{j>i} \phi_j(\mathbf{y})\right] \leq 0 \; \forall i. \tag{5.26}$$

In other words, we would require at every position that the expected number of nouns after that position is at least as large as the probability that the label at that position is a determiner. As before, this will not ensure that with probability 1 every determiner will be followed by a noun. To encode the statement with high probability rather than in expectation would break the model's Markov property, since the model would need to keep track

of whether there has been a determiner at position i in order to decide whether a noun is required at the last position.

Similarly, we can encode that the expected number of nouns in a sentence should be at least as large as the expected number of determiners. The same constraint can also be encoded for the entire corpus: we can say that there should be at least as many nouns in the corpus as there are verbs, while allowing some sentences to have more verbs than nouns. Table 5.2 contains some other examples of constraints that can be encoded using factorizable constraint features.

5.6 Trees

Table 5.3 shows some examples of the kind of information that is easy to encode about dependency trees. Two models that might be able to benefit from the constraints presented in Table 5.3 are described in Chapter 8. Both produce a projective dependency representation. However, Druck et al. [2009] describe a non-projective model that might potentially benefit from the addition of a constraint that the trees should be mostly projective. The important property of all these models is that they define a probability distribution over parse trees as a product over edge factors:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\theta(\mathbf{x})} \prod_{(i,j)\in\mathbf{y}} \psi_\theta(i, j, \mathbf{x}), \tag{5.27}$$

where $\mathbf{y}$ represents a tree composed of edges (i, j) with head node i and child node j. For a non-projective parser, $p_\theta(\mathbf{y}|\mathbf{x})$ is normalized so that all directed trees are included and can have non-zero probability. For a projective parser, only trees that do not have crossing edges have non-zero probability and the distribution is normalized over these only. In both cases, our constraint features ϕ must decompose as sums of edge features in order to meet Definition 5.1. Table 5.3 summarizes a few decomposable constraints for dependency trees. The constraints in Table 5.3 are defined in terms of basic edge features:

$$\phi_{i,j}(\mathbf{y}) = \begin{cases} 1 & \text{if} (i, j) \in \mathbf{y} \\ 0 & \text{otherwise.} \end{cases} \tag{5.28}$$

Because the definitions of $\mathcal{Q}$ in each row of Table 5.3 uses linear combinations of $\phi_{i,j}$, they are decomposable constraints according to Definition 5.1. Table 5.3 is intended to give some simple examples, and particularly to impart some intuition about the kinds of statements that can be made in the language of expectation constraints on decomposable features.

All of the constraints in Table 5.3 have corpus-wide versions that could also be encoded, so for example rather than requiring that each sentence be mostly left-branching, we could make a similar requirement of the entire corpus and allow some sentences to be right-branching as long as overall the corpus is mostly left-branching.

Table 5.4 summarizes some constraints that are not decomposable with respect to a model of the form of Equation 5.27. Note that even though they are not decomposable, that does not mean that there is efficient projection is possible. It simply means that we cannot directly apply Proposition 2.1 in order to compute the KL-projection. For example, under a modified dynamic program siblings, and grandchild constraints are possible [Eisner, 1996, McDonald, 2006], and we describe a procedure for projecting onto the almost projective constraint. The rest of this section explains why the constraints in Table 5.4 are not decomposable.

5.6.1 Depth and Branching Factor

As shown in Table 5.3, it is straightforward to encode a constraint that requires two edges to both be in all parse trees (with probability 1). It is also straightforward to encode a constraint that requires at least one of two edges to be on, at least in expectation. So a conjunction and a disjunction can each be encoded using decomposable constraint features. However, it is not possible to encode a disjunction of conjunctions in a decomposable manner. If we could encode this, it would allow us to capture grandchild, sibling, depth and branching factor relationships. In order to prove this, we will use the contrapositive of Proposition 2.2: we will show that the minimizer $q^*(\mathbf{y})$ of the KL divergence to $p_\theta(\mathbf{y}|\mathbf{x})$ cannot be represented in a factorized form as in Equation 5.27. Since factorizable constraints lead to a factorizable distribution $q^*(\mathbf{y})$ we can conclude that the constraints cannot be factorizable.

Name	Statement	How to encode it with ϕ
Directed Edge	Token i should be a parent of token j	$\mathcal{Q} = \{q : \mathbf{E}_q[\phi_{i,j}(\mathbf{y})] \geq 1\}$. We can replace "$\geq 1$" by "$\leq 0$" to disallow a particular edge.
Undirected Edge	Token i and j should have some parent/child relationship	$\mathcal{Q} = \{q : \mathbf{E}_q[\phi_{j,i}(\mathbf{y}) + \phi_{i,j}(\mathbf{y})] \geq 1\}$. We can disallow undirected edges by disallowing both directions.
Is Leaf	Token i is a leaf	$\mathcal{Q} = \{q : \mathbf{E}_q[\sum_j \phi_{i,j}(\mathbf{y})] \leq 0\}$. We can state "token i is not a leaf" by using "≥ 1".
Is Root	Token i is the root	$\mathcal{Q} = \{q : \mathbf{E}_q[\sum_j \phi_{j,i}(\mathbf{y})] \leq 0\}$. Is not root can be achieved by using "≥ 1".
Branch Left	More than half of the edges should have the head on the right	$\mathcal{Q} = \left\{q : \mathbf{E}_q\left[\sum_{i>j} \phi_{i,j} - \sum_{i<j} \phi_{i,j}\right] \leq 0\right\}$. Similarly we can enforce more right branching than left branching by changing the direction of the inequality.
Partial Labeling	Tree should contain an α fraction of the labels in some set E	$\mathcal{Q} = \left\{q : \mathbf{E}_q\left[\sum_{(i,j)\in E} \phi_{i,j}(\mathbf{y})\right] \geq \alpha\lvert E\rvert\right\}$. See Chapter 8 for details on experiments with this kind of constraint. It is also possible to require that only a few of certain set of edges are present by changing the "$\geq$" to "$\leq$".

Table 5.3: Summary of simple constraints that are decomposable for trees. In all cases, $\phi_{i,j}(\mathbf{y})$ takes on the value 1 iff the tree $\mathbf{y}$ contains an edge $i \to j$. Hence, $\mathbf{E}_q[\phi_{i,j}(\mathbf{y})]$ is the posterior probability that an edge is present in a tree drawn from q.

Suppose that we have a sentence with the three tokens "A","B" and "C", and our model is parameterized as in Equation 5.27. Let the parameters $\psi_\theta(i, j)$ be 1 for all edges. This means that the distribution $p_\theta(\mathbf{y}|\mathbf{x})$ for a projective parser is the uniform distribution over the trees in Figure 5.3.

Proposition 5.1. *Let* $\mathrm{depth}(\mathbf{y})$ *be the depth of tree* $\mathbf{y}$*. The minimizer* $q^*(\mathbf{y})$ *of*

$$\min_q \quad \mathbf{KL}(q(\mathbf{y})||p_\theta(\mathbf{y}|\mathbf{x})) \quad \text{s.t.} \quad \mathbf{E}_q[\mathrm{depth}(\mathbf{y})] \leq 1 + \epsilon \tag{5.29}$$

where $p_\theta(\mathbf{y}|\mathbf{x})$ *is the uniform distribution over trees in Figure 5.3 does not factorize as Equation 5.27 for any* $0 < \epsilon < 1$*. Consequently the constraint of bounded tree depth is not decomposable.*

Name	Statement	Why we can't encode it
Depth	The tree should have depth at most d	The example in Figure 5.3; see text of Section 5.6.1 and Proposition 5.1 for details.
Branching Factor	Non-leaf nodes in the tree should have a large (or small) degree	The example in Figure 5.3; see text of Section 5.6.1 and Proposition 5.2 for details.
Sibling	Nodes i and j should be siblings in the tree	The example if Figure 5.4 illustrates why this is not possible. See Section 5.6.2, and Proposition 5.3 for details.
Almost Projective	The expectation that a tree chosen according to the mode is projective should be at least ϵ.	The example in Figures 5.4 and 5.5 illustrates why this is not decomposable. **Note** there is a work-around, if we are willing to let $q(\mathbf{y})$ be a mixture model. See the text of Section 5.6.3 for details.
Grandchild	Node i should be the grandparent of node j	The example in Figure 5.6, as described in Section 5.6.4 and Proposition 5.6.

Table 5.4: Summary of some natural constraints that cannot be efficiently enforced in the PR framework given a first-order dependency model (Equation 5.27).

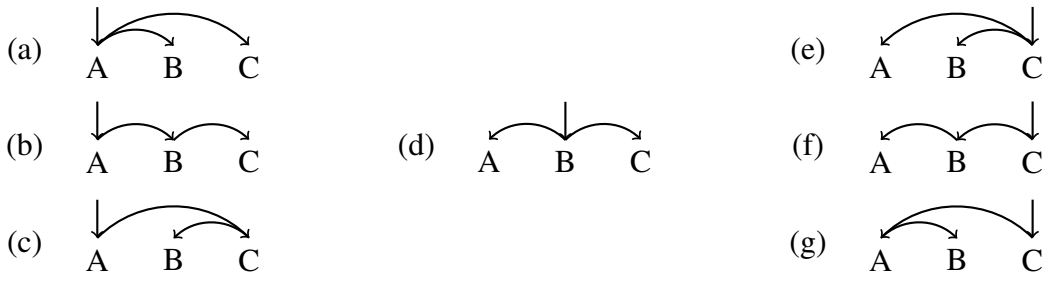


Figure 5.3: An example illustrating why it is not possible to encode grandparent/grandchild, depth and branching factor relationships. Shown are the seven possible projective trees on three vertices. If the parameters are identical for each edge, then each tree occurs with probability $\frac{1}{7}$ under the model. See text for details.

Proof. First observe that trees (a), (d) and (e) in Figure 5.3 have depth 1 while the remaining trees, (b), (c), (f) and (g) have depth 2. Consequently, for $1 > \epsilon > 0$, the minimizer of Equation 5.29 has the same probability $q^*(\mathbf{y}_a)$ for trees (a), (d) and (e) and the same probability $q^*(\mathbf{y}_b)$ for trees (b), (c), (f) and (g) in Figure 5.3.

We can use the fact that trees (a) and (e) have the same set of edges as trees (c) and (g) to show that it is not consistent for trees (a) and (e) to have different probabilities from (c) and (g).

Without loss of generality, let the model $q^*(\mathbf{y})$ be parameterized as $\psi_\theta(i,j) = \exp(\theta_{ij})$. Taking the log of Equation 5.27, we have:

$$\log q^*(\mathbf{y}|\mathbf{X}) = \sum_{(i,j)\in\mathbf{y}} \theta_{ij} - \log Z_\theta. \tag{5.30}$$

Define $\alpha = \log q^*(\mathbf{y}_a|\mathbf{X}) + \log Z_\theta$ and $\beta = \log q^*(\mathbf{y}_b|\mathbf{X}) + \log Z_\theta$. Trees (a), (e), (c) and (g) give us:

$$\theta_{\surd A} + \theta_{AB} + \theta_{AC} = \alpha \tag{5.31}$$

$$\theta_{\surd C} + \theta_{CB} + \theta_{CA} = \alpha \tag{5.32}$$

$$\theta_{\surd A} + \theta_{CB} + \theta_{AC} = \beta \tag{5.33}$$

$$\theta_{\surd C} + \theta_{AB} + \theta_{CA} = \beta \tag{5.34}$$

respectively, where $\surd$ represents the root symbol. By summing Equations 5.31 and 5.32 and subtracting Equations 5.33 and 5.34 we get that $\alpha = \beta$. Consequently, $q^*(\mathbf{y})$ does not factorize as Equation 5.27, and tree depth constraints are not decomposable. □

Note that as $\epsilon \to 0$, $q^*(\mathbf{y}_b) \to 0$. We might potentially be interested in the limit: $\lim \epsilon \to 0$, in which case we might be satisfied if the probabilities of trees (b), (c), (f) and (g) in Figure 5.3 tend to zero, without necessarily being equal to each other. The proof of Proposition 5.1 shows that even this is not possible while maintaining a uniform distribution over trees (a), (d) and (e).

The proof of Proposition 5.1 also shows that it is not possible to lower bound the depth of the tree. Notice, that for trees on only three nodes, the three of depth 2 also have branching factor 1, since they are a chain, while the trees of depth 1 have branching factor 2, since

the only non-leaf node has two children. Consequently, the same proof can be used for the following proposition regarding branching factor.

Proposition 5.2. *Let* $\mathrm{branch}(\mathbf{y})$ *be the branching factor of* $\mathbf{y}$*. That is, the average number of children of non-leaf nodes. The minimizer* $q^*(\mathbf{y})$ *of*

$$\min_q \quad \mathbf{KL}(q(\mathbf{y})||p_\theta(\mathbf{y}|\mathbf{x})) \quad \text{s.t.} \quad \mathbf{E}_q[\mathrm{branch}(\mathbf{y})] \geq 1 + \epsilon \tag{5.35}$$

where $p_\theta(\mathbf{y}|\mathbf{x})$ *is the uniform distribution over trees in Figure 5.3 does not factorize as Equation 5.27 for any* $0 < \epsilon < 1$*. Consequently the constraint of bounded branching factor is not decomposable.*

Note that it is also not possible to upper bound the branching factor.

5.6.2 Siblings

As with depth constraints, sibling constraints do not decompose in the same way as Equation 5.27. The proof is similar to the proof of Proposition 5.1, but uses the example in Figure 5.4.

Proposition 5.3. *Let* $\mathrm{sibling}_{\mathrm{DE}}(\mathbf{y})$ *be a function that takes on value 1 if nodes D and E are siblings in tree* $\mathbf{y}$ *and 0 otherwise. The minimizer* $q^*(\mathbf{y})$ *of*

$$\min_q \quad \mathbf{KL}(q(\mathbf{y})||p_\theta(\mathbf{y}|\mathbf{x})) \quad \text{s.t.} \quad \mathbf{E}_q\left[\mathrm{sibling}_{\mathrm{DE}}(\mathbf{y})\right] \geq \epsilon \tag{5.36}$$

where $p_\theta(\mathbf{y}|\mathbf{x})$ *is the uniform distribution over trees in Figure 5.4 does not factorize as Equation 5.27 for any* $0 < \epsilon < 1$*. Consequently sibling constraints are not decomposable.*

Proof. The proof is very similar to the proof of Proposition 5.1.

Observe that trees (a), (b) and (c) in Figure 5.4 have $\mathrm{sibling}_{\mathrm{DE}}(\mathbf{y}) = 1$ and the remaining trees, (d), (e), (f) have $\mathrm{sibling}_{\mathrm{DE}}(\mathbf{y}) = 0$. Consequently, for $1 > \epsilon > 0$, the minimizer of Equation 5.36 has the same probability $q^*(\mathbf{y}_a)$ for trees (a), (b) and (c) and the same probability $q^*(\mathbf{y}_b)$ for trees (d), (e) and (f) in Figure 5.3.

		parent					
		root	A	B	C	D	E
child	A	1	-	-	-	-	-
	B	-	1	-	-	-	-
	C	-	-	1	-	-	-
	D	-	1	1	1	-	-
	E	-	1	1	1	-	-

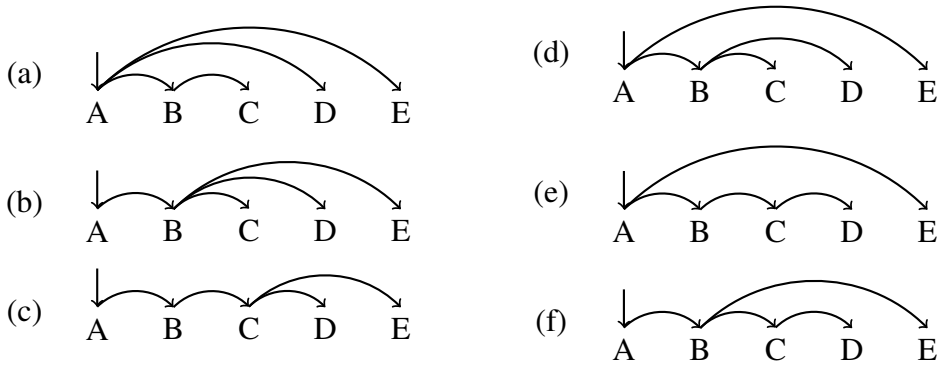


Figure 5.4: An example illustrating why it is not possible to encode sibling relationships. In the top panel are the parameters of a dependency. Dashes represent a zero factor; the bottom panel shows the six projective trees with non-zero probability. Each has probability $\frac{1}{6}$ under the model. See text for details.

Using the same notation as in Proposition 5.1, Let α and β be defined as:

$$\alpha = \log q^*(\mathbf{y}_a) + \log Z_\theta - \theta_{\surd A} - \theta_{AB} - \theta_{BC} \tag{5.37}$$

$$\beta = \log q^*(\mathbf{y}_b) + \log Z_\theta - \theta_{\surd A} - \theta_{AB} - \theta_{BC} \tag{5.38}$$

By taking the log of Equation 5.27 and substituting α and β, for trees (b–f) respectively, we have:

$$\theta_{BD} + \theta_{BE} = \alpha \tag{5.39}$$

$$\theta_{CD} + \theta_{CE} = \alpha \tag{5.40}$$

$$\theta_{BD} + \theta_{AE} = \beta \tag{5.41}$$

$$\theta_{CD} + \theta_{AE} = \beta \tag{5.42}$$

$$\theta_{CD} + \theta_{BE} = \beta. \tag{5.43}$$

Equations 5.41 and 5.42 give us that $\theta_{BD} = \theta_{CD}$. This combined with Equations 5.39 and 5.40 gives us that $\theta_{BE} = \theta_{CE}$. Combining this with Equations 5.40 and 5.43 we get that $\alpha = \beta$. Consequently, $q^*(\mathbf{y})$ does not factorize as Equation 5.27, and sibling constraints are not decomposable. □

Note that Proposition 5.3 does not make a claim about $\epsilon = 1$. It is easy to see that if trees (a-c) have to each have non-zero probability, then trees (d-f) must also have non-zero probability because each edge in trees (d-f) appears in trees (a-c). Similarly for $\epsilon = 0$.

In practice, it might be sufficient to approximate $q^*(\mathbf{y})$ as long as the approximation is good. If we are only interested in approximating $q^*(\mathbf{y})$ as $\epsilon \to 1$, then it might be possible to find decomposable constraints that have a solution that approaches $q^*(\mathbf{y})$ as $\epsilon \to 1$. In this case, it would not necessarily be the case that trees (d-f) have the same probability, only that their probability is small and that trees (a-c) have approximately uniform probability. We have not been able to find decomposable constraints that achieve this kind of approximation, and it is possible that a more detailed analysis will show they do not exist.

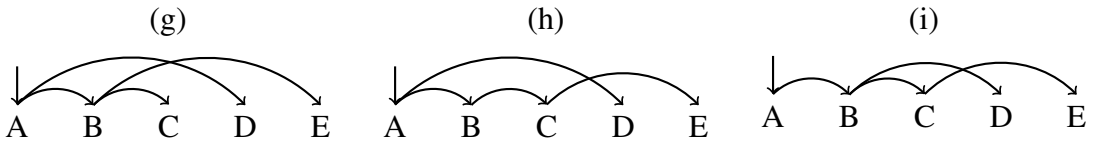


Figure 5.5: The additional non-projective tree possible by the parameters in Figure 5.4. The numbering continues from Figure 5.4. The probability of each tree for the non-projective parser is $\frac{1}{9}$.

5.6.3 Almost Projective

If we have a model that allows non-projective trees, then the parameters in Figure 5.4 allow the tree in Figure 5.5 in addition to the trees in Figure 5.4. In that case, the distribution $p_\theta(\mathbf{y})$ will be uniform over the nine trees. There are many languages that allow non-projective trees, but where trees are mostly projective. For such a language we might want to control the probability that a tree drawn from the model over the entire corpus will be non-projective. The following proposition shows that such constraints are not decomposable, however see Proposition 5.5 for a workaround to this problem.

Proposition 5.4. *Let* $\text{proj}(\mathbf{y})$ *be a function that takes on value 1 if* $\mathbf{y}$ *is a projective tree and 0 otherwise. The minimizer* $q^*(\mathbf{y})$ *of*

$$\min_q \quad \mathbf{KL}(q(\mathbf{y})||p_\theta(\mathbf{y}|\mathbf{x})) \quad \text{s.t.} \quad \mathbf{E}_q[\text{proj}(\mathbf{y})] \geq \epsilon \tag{5.44}$$

where $p_\theta(\mathbf{y}|\mathbf{x})$ *is the uniform distribution over the nine trees in Figures 5.4 and 5.5 does not factorize as Equation 5.27 for any* $0 < \epsilon < 1$*. Consequently almost-projective constraints are not decomposable.*

Proof. The proof is very similar to the proof of Propositions 5.1 and 5.3. For $1 > \epsilon > 0$, the minimizer of Equation 5.44 has the same probability $q^*(\mathbf{y}_a)$ for trees (a-f) in Figure 5.4 and the same probability $q^*(\mathbf{y}_b)$ for trees (g-i) Figure 5.5. Using the notation from the proof of Proposition 5.3, with α corresponding to $q^*(\mathbf{y}_a)$ and β to $q^*(\mathbf{y}_b)$ we have for trees (b)

and (g-i) respectively:

$$\theta_{BD} + \theta_{BE} = \alpha \tag{5.45}$$

$$\theta_{AD} + \theta_{BE} = \beta \tag{5.46}$$

$$\theta_{AD} + \theta_{CE} = \beta \tag{5.47}$$

$$\theta_{BD} + \theta_{CE} = \beta. \tag{5.48}$$

Equations 5.46 and 5.47 give us that $\theta_{BE} = \theta_{CE}$. This together with Equations 5.45 and 5.48 allow us to conclude that $\alpha = \beta$. Consequently, $q^*(\mathbf{y})$ does not factorize as Equation 5.27, and almost-projective constraints are not decomposable. □

Even though the almost-projective constraint is not decomposable, we can represent the minimizer $q^*(\mathbf{y})$ as a mixture of projective and non-projective distributions.

Concretely, suppose that $p_\theta(\mathbf{y})$ is a non-projective model defined by defined by Equation 5.27, and define $p_\theta^{\text{proj}}(\mathbf{y})$ to be the projective model defined by the same equation. The difference between $p_\theta(\mathbf{y})$ and $p_\theta^{\text{proj}}(\mathbf{y})$ is normalization and the set of $\mathbf{y}$ for which the probability is non-zero. The projection we would like to perform is

$$\min_q \ \mathbf{KL}(q(\mathbf{y})||p_\theta(\mathbf{y})) \quad \text{s.t.} \ \ \mathbf{E}_q[\text{proj}(\mathbf{y})] \geq \epsilon. \tag{5.49}$$

The following proposition can help us to solve the projection.

Proposition 5.5. *The solution $q^*(\mathbf{y})$, to the projection in Equation 5.49 has the form of a mixture model between the projective and non-projective models:*

$$q^*(\mathbf{y}) = (1-\alpha)p_\theta(\mathbf{y}) + \alpha p_\theta^{\text{proj}}(\mathbf{y}), \tag{5.50}$$

where $0 \leq \alpha \leq 1$ is the smallest real sufficient to ensure the constraint that $\mathbf{E}_q[\text{proj}(\mathbf{y})] \geq \epsilon$. Specifically $\alpha = \frac{\epsilon - \mathbf{E}_{p_\theta}[\text{proj}(\mathbf{y})]}{1 - \mathbf{E}_{p_\theta}[\text{proj}(\mathbf{y})]}$ if $\epsilon \geq \mathbf{E}_{p_\theta}[\text{proj}(\mathbf{y})]$ and $\alpha = 0$ otherwise.

Proof. From Proposition 2.1 we have that $q^*(\mathbf{y}) \propto p_\theta(\mathbf{y}) \exp(-\lambda^* \text{proj}(\mathbf{y}))$. [1] Splitting

[1] The dual parameters will have the constraint $\lambda \leq 0$ because we have $\geq \epsilon$ rather than $\leq \epsilon$ as in Proposition 2.1.

this out into cases where $\mathrm{proj}(\mathbf{y}) = 1$ and where $\mathrm{proj}(\mathbf{y}) = 0$, we have

$$q^*(\mathbf{y}) \propto p_\theta(\mathbf{y}) e^{-\lambda^* \mathrm{proj}(\mathbf{y})} = \begin{cases} e^{-\lambda} p_\theta(\mathbf{y}) & \text{if } \mathbf{y} \text{ is projective} \\ p_\theta(\mathbf{y}) & \text{if } \mathbf{y} \text{ is not projective.} \end{cases} \tag{5.51}$$

Because $\lambda \leq 0$, we can express Equation 5.51 as $q^*(\mathbf{y}) \propto p_\theta^{\text{nop}}(\mathbf{y}) + (1 - e^{-\lambda}) p_\theta^{\text{proj}}(\mathbf{y})$, which gives us Equation 5.50 with $\alpha = (1 - e^{-\lambda})$ as desired. The value of α is obtained by observing that under the projective model all trees are projective, and algebraic manipulation of Equation 5.50. □

Proposition 5.5 tells us how to compute the projection onto the set of constraints $\mathbf{E}_q[\mathrm{proj}(\mathbf{y})] \geq \epsilon$, however in order to use it, we have to be able to estimate $\mathbf{E}_{p_\theta}[\mathrm{proj}(\mathbf{y})]$. If we are interested in a corpus-wide constraint we can use samples or the Viterbi parses of the corpus in order to estimate this value.

5.6.4 Grandparents and Grandchildren

Proposition 5.6. *Let* $\mathrm{grandparent}_{\mathrm{AE}}(\mathbf{y})$ *be a function that takes on value 1 if node A is a grandparent of node E in tree* $\mathbf{y}$ *and 0 otherwise. The minimizer* $q^*(\mathbf{y})$ *of*

$$\min_q \quad \mathbf{KL}(q(\mathbf{y}) || p_\theta(\mathbf{y}|\mathbf{x})) \quad \text{s.t.} \quad \mathbf{E}_q\left[\mathrm{grandparent}_{\mathrm{AE}}(\mathbf{y})\right] \geq \epsilon \tag{5.52}$$

where $p_\theta(\mathbf{y}|\mathbf{x})$ *is the uniform distribution over trees in Figure 5.6 does not factorize as Equation 5.27 for any* $0 < \epsilon < 1$. *Consequently grandparent constraints are not decomposable.*

Proof. The proof is very similar to the proof of Propositions 5.1, 5.3 and 5.4. For $1 > \epsilon > 0$, the minimizer of Equation 5.44 has the same probability $q^*(\mathbf{y}_a)$ for trees (a) and (c) in Figure 5.6 and the same probability $q^*(\mathbf{y}_b)$ for trees (b), (d) and (e). Using the same notation as in the proof of Proposition 5.3, let α correspond to $q^*(\mathbf{y}_a)$ and β correspond to

		parent					
		root	A	B	C	D	E
child	A	1	-	-	-	-	-
	B	-	1	-	-	-	-
	C	-	1	1	-	-	-
	D	-	-	-	1	-	-
	E	-	-	1	1	1	-

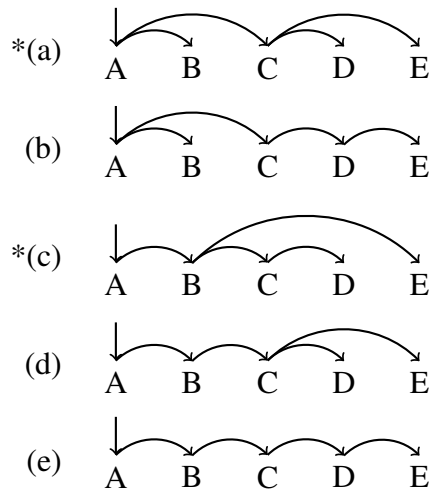


Figure 5.6: An example illustrating why it is not possible to encode grandparent relationships. In the left panel are the parameters of a dependency model. Dashes represent a zero factor; the right panel shows the five projective trees with non-zero probability. Each has probability $\frac{1}{5}$ under the model. See text for details.

$q^*(\mathbf{y}_b)$. We then have from trees (a), (b), (d) and (e) respectively:

$$\theta_{AC} + \theta_{CE} = \alpha \tag{5.53}$$

$$\theta_{AC} + \theta_{DE} = \beta \tag{5.54}$$

$$\theta_{BC} + \theta_{CE} = \beta \tag{5.55}$$

$$\theta_{BC} + \theta_{DE} = \beta. \tag{5.56}$$

Equations 5.55 and 5.56 imply $\theta_{CE} = \theta_{DE}$. This along with Equations 5.53 and 5.54 implies $\alpha = \beta$. Consequently, the optimizer of Equation 5.52 does not factorize as Equation 5.27, and grandparent constraints are not factorizable. □

As with sibling constraints we have not dealt with the case $\epsilon = 1$. In this case, we want to encode the uniform distribution over trees (a) and (c), in the form of Equation 5.27. It is easy to see that this is not possible, since tree (d) in Figure 5.6 is composed of edges in trees (a) and (c). Hence if trees (a) and (c) have non-zero probability, tree (d) must also have non-zero probability. As with sibling constraints, we are not making a claim about distributions that only approximate $q^*(\mathbf{y})$, although a more detailed analysis might reveal a stronger statement that includes a claim about approximations.

Part II

Empirical Study

Chapter 6

Statistical Word Alignments

Word alignments, introduced by Brown et al. [1994] as hidden variables in probabilistic models for statistical machine translation (IBM models 1-5), describe the correspondence between words in source and target sentences. We will denote each target sentence as $\mathbf{x}^t = (x_1^t, \ldots, x_i^t, \ldots, x_I^t)$ and each source sentence as $\mathbf{x}^s = (x_1^s, \ldots, x_j^s, \ldots, x_J^s)$. A word alignment will be represented as a matrix with entries y_{ij} indicating that target word i is a translation of source word j. In cases where the model generating the alignment is a hidden Markov model, we will use the conventional HMM notation for hidden variables $y_i = j$ to mean $y_{ij} = 1$ and $y_{ij'} = 0 \; \forall j' \neq j$. Although the original IBM models are no longer competitive for machine translation, the resulting word alignments are still a valuable resource. Word alignments are used primarily for extracting minimal translation units for machine translation, for example, phrases in phrase-based translation systems [Koehn et al., 2003] and rules in syntax-based machine translation [Galley et al., 2004, Chiang et al., 2005], as well as for MT system combination [Matusov et al., 2006]. But their importance has grown far beyond machine translation: for instance, transferring annotations between languages by projecting POS taggers, NP chunkers and parsers through word alignment [Yarowsky and Ngai, 2001, Rogati et al., 2003, Hwa et al., 2005, Ganchev et al., 2009a]; discovery of paraphrases [Bannard and Callison-Burch, 2005, Callison-Burch, 2007, 2008]; and joint unsupervised POS and parser induction across languages [Snyder and Barzilay, 2008, Snyder et al., 2009b].

Here we describe two types of prior knowledge that when introduced as constraints in

different word alignment models significantly boost their performance. The two constraints are: (i) bijectivity: "one word should not translate to many words"; and (ii) symmetry: "directional alignments of one model should agree with those of another model". A more extensive description of these constraints applied to the task of word alignments and the quality of the resulting alignments can be found in Graça et al. [2009b].

6.1 Models

We consider two models below: IBM Model 1 proposed by Brown et al. [1994] and the HMM model proposed by Vogel et al. [1996]. Both models can be expressed as:

$$p(\mathbf{x}^t, \mathbf{y} \mid \mathbf{x}^s) = \prod_j p_d(y_j \mid j, y_{j-1}) p_t(\mathbf{x}^t_j \mid \mathbf{x}^s_{y_j}), \tag{6.1}$$

where $\mathbf{y}$ is the alignment and y_j is the index of the hidden state (source language index) generating the target language word at index j. The models differ in their definition of the distortion probability $p_d(y_j \mid j, y_{j-1})$. Model 1 assumes that the target words are generated independently and assigns uniform distortion probability. The HMM model assumes that only the distance between the current and previous source word index is important $p_d(y_j \mid j, y_{j-1}) = p_d(y_j \mid y_j - y_{j-1})$. Both models are augmented by adding a special "null" word to the source sentence.

The likelihood of the corpus, marginalized over possible alignments, is concave for Model 1, but not for the HMM model [Brown et al., 1994, Vogel et al., 1996]. For both models though, standard training using the Expectation Maximization algorithm [Dempster et al., 1977] seeks model parameters θ that maximize the log-likelihood of the parallel corpus.

On the positive side, both models are simple and complexity of inference is $O(I \times J)$ for IBM Model 1 and $O(I \times J^2)$ for the HMM. However there are several problems with the models that arise from their directionality.

- **Non-bijective:** Multiple target words can align to a single source word with no penalty.

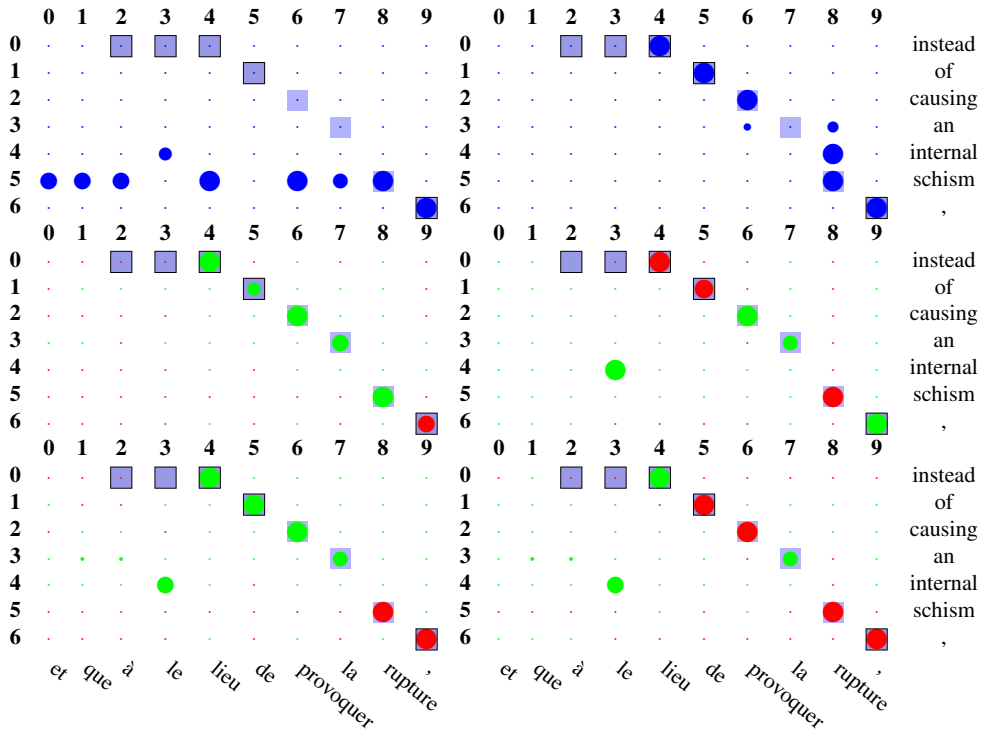


Figure 6.1: Posterior distributions on an English to French sentence using the HMM model. [1] **Left:** EN→FR model. **Right:** FR→ EN model. **Top:** Regular EM posteriors. **Middle:** After applying bijective constraint. **Bottom:** After applying symmetric constraint. Squares are human annotations, circles are output from the model. *Sure* alignments are squares with borders; *possible* alignments are squares without borders. Circle size indicates probability value. Circle color in the middle and bottom rows indicates difference in posterior from the top row. Green - higher probability, red - lower probability.

- **Asymmetric:** Swapping the source and target languages can produce very different alignments, since only constraints and correlations between consecutive positions on one side are enforced by the models.

The top row of Figure 6.1 shows an example of the posterior distribution for the alignment between an English and a French sentence using the HMM model. [1] The left figure shows the alignment in the English to French direction where the rows are source words and columns are target words, while the right figure shows the alignment posteriors of the opposite direction. The first observation we make is that the posteriors are concentrated around particular source words (rare words occurring less than 5 times in the corpus) in

[1] For simplicity of exposition, in this chapter we have not discussed details such as the inclusion of a special *null* word among the set of states, and we do not show the null word in Figure 6.1. For example, with both bijective and symmetric constraints, the French words "et", "que" and "à" are generated by the null word, and hence remain unaligned.

both directions, instead of being spread across different words. This is a well known problem when training using EM, called the "garbage collector effect" [Brown et al., 1993]. That is, rare words in the source language end up aligned to too many words in the target language because the generative model has to distribute translation probability for each source word among all candidate target words. Since the rare source word occurs in only a few sentences it needs to spread its probability mass over fewer competing target words. In this case, choosing to align the rare word to all of these target words leads to higher likelihood than correctly aligning them or aligning them to the special *null* word, since it increases the likelihood of this sentence without lowering the likelihood of many other sentences.

6.2 Bijectivity Constraints

Bijectivity constraints are based on the observation that in most gold alignments, words are aligned one-to-one. We would like to introduce this trend into the model, but adding this to the model directly breaks the Markov property. In fact, summing over one-to-one or near one-to-one weighted matchings is a classical #P-Complete problem [Valiant, 1979]. However, introducing alignment degree constraints *in expectation* in the PR framework is easy and tractable. We simply add inequality constraints $\mathbf{E}[\phi(\mathbf{x}, \mathbf{y})] \leq 1$ where we have one feature for each source word j that counts how many times it is aligned to a target word in the alignment $\mathbf{y}$:

$$\textbf{Bijective Features}: \qquad \phi_j(\mathbf{x}, \mathbf{y}) = \sum_i \mathbf{1}(y_i = j).$$

For example, in the alignment at the top right of Figure 6.1, the posteriors over the source word *schism* clearly sum to more than 1. The effect of applying PR constraints to the posteriors is shown in the second row. Enforcing the one to (at most) one constraint clearly alleviates the garbage collector effect. Moreover, when distributing the probability mass to the other words, most of the probability mass goes into the correct positions (as can be seen by comparison to the gold alignments).

6.3 Symmetry Constraints

Word alignment should not depend on translation direction, but this principle is clearly violated by the directional models. In fact, each directional model makes different mistakes. The standard approach is to train two models independently and then intersect their predictions [Och and Ney, 2003]. However, we show that it is much better to train two directional models concurrently, coupling their posterior distributions over alignments with constraints that force them to approximately agree. The idea of training jointly has also been explored by Matusov et al. [2004] and Liang et al. [2006], although their formalization is quite different.

Let the directional models be defined as: $\overrightarrow{p}(\overrightarrow{\mathbf{y}})$ (forward) and $\overleftarrow{p}(\overleftarrow{\mathbf{y}})$ (backward). We suppress dependence on $\mathbf{x}^s$ and $\mathbf{x}^t$ for brevity. Define $\mathbf{y}$ to range over the union of all possible directional alignments $\overrightarrow{\mathbf{y}} \cup \overleftarrow{\mathbf{y}}$. We then define a mixture model $p(\mathbf{y}) = \frac{1}{2}\overrightarrow{p}(\mathbf{y}) + \frac{1}{2}\overleftarrow{p}(\mathbf{y})$ where $\overleftarrow{p}(\overrightarrow{\mathbf{y}}) = 0$ and vice-versa (i.e., the alignment of one directional model has probability zero according to the other model). We then define the following feature for each target-source position pair i, j:

$$\textbf{Symmetric Features}: \quad \phi_{ij}(\mathbf{x}, \mathbf{y}) = \begin{cases} +1 & \mathbf{y} \in \overrightarrow{\mathbf{y}} \text{ and } \overrightarrow{y_i} = j \\ -1 & \mathbf{y} \in \overleftarrow{\mathbf{y}} \text{ and } \overleftarrow{y_j} = i \\ 0 & \text{otherwise} \end{cases}.$$

The feature takes the value zero in expectation if a word pair i, j is aligned with equal probability in both directions. So the constraint we want to impose is $\mathbf{E}_q[\phi_{ij}(\mathbf{x}, \mathbf{y})] = 0$ (possibly with some small violation). Note that this constraint is only feasible if the posteriors are bijective. Clearly these features are fully factored, so to compute expectations of these features under the model q we only need to be able to compute them under each directional model, as we show below. To see this, we have by the definition of q_λ and p_θ,

$$q_\lambda(\mathbf{y} \mid \mathbf{x}) = \frac{\overrightarrow{p}(\mathbf{y} \mid \mathbf{x}) + \overleftarrow{p}(\mathbf{y} \mid \mathbf{x})}{2} \frac{\exp\{-\lambda \cdot \phi(\mathbf{x}, \mathbf{y})\}}{Z} = \frac{\overrightarrow{q}(\mathbf{y} \mid \mathbf{x}) \frac{Z_{\overrightarrow{q}}}{\overrightarrow{p}(\mathbf{x})} + \overleftarrow{q}(\mathbf{y} \mid \mathbf{x}) \frac{Z_{\overleftarrow{q}}}{\overleftarrow{p}(\mathbf{x})}}{2Z}, \tag{6.2}$$

where we have defined:

$$\overrightarrow{q}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z_{\overrightarrow{q}}} \overrightarrow{p}(\mathbf{y}, \mathbf{x}) \exp\{-\lambda \cdot \phi(\mathbf{x}, \mathbf{y})\} \quad \text{with} \quad Z_{\overrightarrow{q}} = \sum_{\mathbf{y}} \overrightarrow{p}(\mathbf{y}, \mathbf{x}) \exp\{-\lambda \cdot \phi(\mathbf{x}, \mathbf{y})\},$$

$$\overleftarrow{q}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z_{\overleftarrow{q}}} \overleftarrow{p}(\mathbf{y}, \mathbf{x}) \exp\{-\lambda \cdot \phi(\mathbf{x}, \mathbf{y})\} \quad \text{with} \quad Z_{\overleftarrow{q}} = \sum_{\mathbf{y}} \overleftarrow{p}(\mathbf{y}, \mathbf{x}) \exp\{-\lambda \cdot \phi(\mathbf{x}, \mathbf{y})\},$$

$$Z = \frac{1}{2}\left(\frac{Z_{\overrightarrow{q}}}{\overrightarrow{p}(\mathbf{x})} + \frac{Z_{\overleftarrow{q}}}{\overleftarrow{p}(\mathbf{x})}\right).$$

All these quantities can be computed separately in each model.

The last row in Figure 6.1 shows both directional posteriors after imposing the symmetric constraint. Note that the projected posteriors are equal in the two models. Also, one can see that in most cases the probability mass was moved to the correct place with the exception of the word pair *internal/le*; this is because the word *internal* does not appear on the French side, but the model still has to spread around the probability mass for that word. In this case the model decided to accumulate it on the word *le* instead of moving it to the *null* word.

6.4 Results

We evaluated the constraints using the Hansards corpus [Och and Ney, 2000] of English/French. Following prior work by Och and Ney [2003], we initialize the Model 1 translation table with uniform probabilities over word pairs that occur together in same sentence. The HMM is initialized with the translation probabilities from Model 1 and with uniform distortion probabilities. We train M1 for 5 iterations and train the HMM model until no further improvement on precision and recall is seen on standard (small) development set for this corpus. We note that when using regular EM training this requires around 4 iterations, while just 2 iterations suffices when using PR. This is likely due to the added information that the constraints provide. We use a 40 word maximum length cutoff for training sentences and train all models on $100,000$ sentences, testing precision and recall on the standard test set.

Figure 6.4 shows the precision vs recall curves of both models (EN-FR, FR-EN independently) when training using standard EM versus PR with both constraints, and the re-

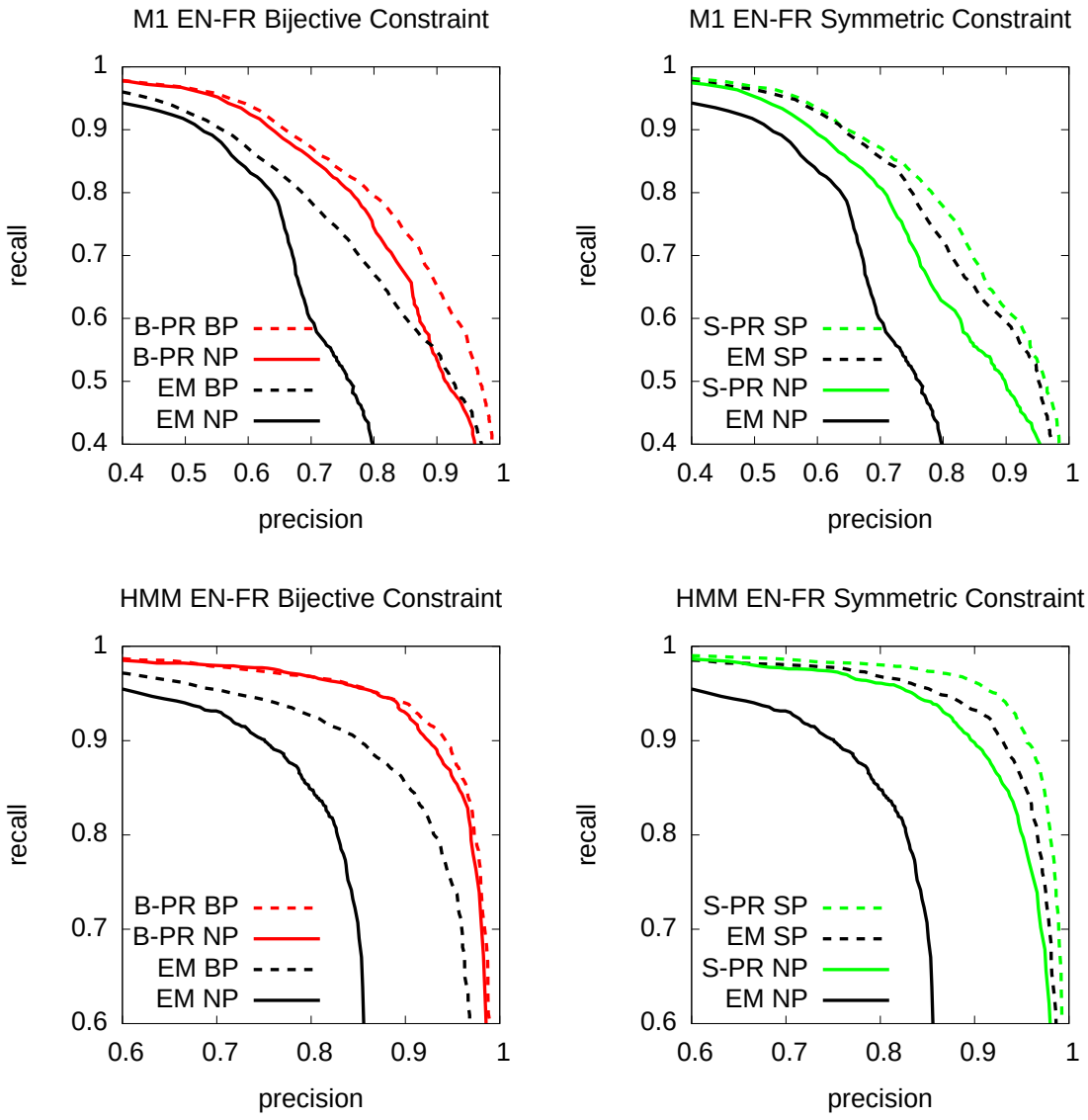


Figure 6.2: Precision vs Recall curves of both models using standard EM training (Regular) versus PR with bijective constraints (Bijective) and symmetry constraints (Symmetric) and different decoding types: decoding without any projection (NP), doing bijective projection before decoding (BP), and doing symmetric projection before decoding (SP). Data is 100k sentences of the Hansards corpus. Highest label in the legend corresponds to highest line in the graph, second highest label to second highest line, and so on.

sults of additionally applying the constraints at decode time in order to tease apart the effect of the constraints during training vs. during testing. The first observation is that training with PR significantly boosts the performance of each model. Moreover using the projection at decode time always increases performance. Comparing both constraints, it seems that bijective is more useful at training time. Note that using this constraint at decode time with regular training yields worse results than just training with the same constraint using PR. On the other hand, the symmetric constraint is stronger at decode time.

Chapter 7

Mutli-view learning

This chapter is based on Ganchev et al. [2009b, 2008, 2010]. Multi-view learning refers to a set of semi-supervised methods which exploit redundant views of the same input data [Blum and Mitchell, 1998, Collins and Singer, 1999, Brefeld et al., 2005, Sindhwani et al., 2005]. These multiple views can come in the form of context and spelling features in the case of text processing and segmentation, hypertext link text and document contents for document classification, and multiple cameras or microphones in the case of speech and vision. Multi-view methods typically begin by assuming that each view alone can yield a good predictor. Under this assumption, we can regularize the models from each view by constraining the amount by which we permit them to disagree on unlabeled instances. This regularization can lead to better convergence by significantly decreasing the effective size of our hypothesis class [Balcan and Blum, 2005, Kakade and Foster, 2007, Rosenberg and Bartlett, 2007]. This idea is related to the symmetry constraints described in Chapter 6.

In this chapter, we use PR to derive a multi-view learning algorithm. The idea is very simple: train a model for each view, and use constraints that the models should agree on the label distribution. Where our work is most similar to co-regularization schemes, a minimum Kullbeck-Leibler (KL) distance projection can be computed in closed form resulting in an algorithm that performs better than both CoBoosting and two view Perceptron on several natural language processing tasks. In this case, the resulting regularizer is identical to adding a penalty term based on the Bhattacharyya distance [Kailath, 1967] between models trained using different views.

In addition, this framework allows us to use different labeled training sets for the two classifiers, in the case where they have different label sets. That is, we don't require that our two views are both on the same labeled corpus. In that case, we can reduce the hypothesis space by preferring pairs of models that agree on *compatible* labeling of some additional unlabeled data rather than on *identical* labeling, while still minimizing KL in closed form. When the two views come from models that differ not only in the label set but also in the model structure of the output space, our framework can still encourage agreement, but the KL minimization cannot be computed in closed form. Finally, this method uses soft assignments to latent variables resulting in a more stable optimization procedure.

7.1 Stochastic Agreement

Note that the constraint in this chapter is similar to the one described in Chapter 6, but here we focus on discriminative learning and the formulation is slightly different. For notational convenience, we focus on two view learning in this exposition, however the generalization to more than two views is fairly straightforward. Also, we focus on two discriminative log-linear models and start by considering the setting of complete agreement. In this setting we have a common desired output for the two models and we believe that each of the two views is sufficiently rich to predict labels accurately. We can leverage this knowledge by restricting our search to model pairs p_1,p_2 that satisfy $p_1(\mathbf{y} \mid \mathbf{x}) \approx p_2(\mathbf{y} \mid \mathbf{x})$. Since p_1 and p_2 each define a distribution over labels, we will consider the product distribution $p_1(\mathbf{y}_1)p_2(\mathbf{y}_2)$ and define constraint features such that our proposal distribution $q(\mathbf{y}_1, \mathbf{y}_2)$ will have the same marginal for $\mathbf{y}_1$ and $\mathbf{y}_2$. In particular, we will have one constraint feature for each label $\mathbf{y}$:

$$\phi_{\mathbf{y}}(\mathbf{y}_1, \mathbf{y}_2) = \delta(\mathbf{y}_1 = \mathbf{y}) - \delta(\mathbf{y}_2 = \mathbf{y}) \tag{7.1}$$

Where $\delta(\cdot)$ is the 0-1 indicator function. The constraint set $\mathcal{Q} = \{q : \mathbf{E}_q[\phi] = 0\}$ will require that the marginals over the two output variables are identical $q(\mathbf{y}_1) = q(\mathbf{y}_2)$. It will be useful in the sequel to define an agreement between two models $\mathbf{agree}(p_1, p_2)$ as

$$\mathbf{agree}(p_1, p_2) = \arg\min_q \ \mathbf{KL}(q(\mathbf{y}_1, \mathbf{y}_2)||p_1(\mathbf{y}_1)p_2(\mathbf{y}_2)) \quad \text{s.t.} \quad \mathbf{E}_q[\phi] = 0 \tag{7.2}$$

Proposition 7.1 relates the Bhattacharyya regularization term to the value of the optimization problem in Equation 7.2. The Bhattacharyya distance is a very natural, symmetric measure of difference between distributions which has been used in many signal detection applications [Kailath, 1967]. It is also related to the well-known Hellinger distance.

Proposition 7.1. *The Bhattacharyya distance* $-\log \sum_{\mathbf{y}} \sqrt{p_1(\mathbf{y})p_2(\mathbf{y})}$ *is equal to* $\frac{1}{2}$ *of the value of the convex optimization problem*

$$\begin{array}{ll} \min\limits_{q \in \mathcal{Q}} & \mathbf{KL}(q(\mathbf{y}_1, \mathbf{y}_2)||p_1(\mathbf{y}_1)p_2(\mathbf{y}_2)) \\ \text{where} & \mathcal{Q} = \{q : \mathbf{E}_q[\delta(\mathbf{y}_1 = \mathbf{y}) - \delta(\mathbf{y}_2 = \mathbf{y})] = 0 \; \forall y\}, \end{array} \tag{7.3}$$

and where $\delta(cond)$ *is* 1 *if cond is true and* 0 *otherwise. Furthermore, the minimizer decomposes as* $q(\mathbf{y}_1, \mathbf{y}_2) = q_1(\mathbf{y}_1)q_2(\mathbf{y}_2)$ *and is given by* $q_i(y) \propto \sqrt{p_1(\mathbf{y})p_2(\mathbf{y})}$.

Proof. The set $\mathcal{Q}$ can equivalently be defined as

$$\mathcal{Q} = \{q(\mathbf{y}_1, \mathbf{y}_2) : \mathbf{E}_q[\phi(\mathbf{y}_1, \mathbf{y}_2)] = 0]\} \tag{7.4}$$

where $\phi(\mathbf{y}_1, \mathbf{y}_2)$ is a vector of features of the form $\delta(\mathbf{y}_1 = \mathbf{y}) - \delta(\mathbf{y}_2 = \mathbf{y})$ with one entry for each possible label $\mathbf{y}$. By a version of Proposition 2.1 for equality constraints, the dual of Equation 7.3 is

$$\arg\max_{\lambda} -\log \sum_{\mathbf{y}_1, \mathbf{y}_2} p(\mathbf{y}_1, \mathbf{y}_2) \exp(\lambda \cdot \phi) \tag{7.5}$$

with $q(\mathbf{y}_1, \mathbf{y}_2) \propto p(\mathbf{y}_1, \mathbf{y}_2) \exp(\lambda \cdot \phi(\mathbf{y}_1, \mathbf{y}_2))$. Noting that the features decompose into $\phi'(\mathbf{y}_1) - \phi'(\mathbf{y}_2)$, we know that $q(\mathbf{y}_1, \mathbf{y}_2)$ decomposes as $q_1(\mathbf{y}_1)q_2(\mathbf{y}_2)$. Furthermore, our constraints require that $q_1(\mathbf{y}) = q_2(\mathbf{y}) \forall \mathbf{y}$. To emphasize that $q_1 = q_2$ we drop the subscript on $q(\mathbf{y})$. We have

$$q(\mathbf{y}_1)q(\mathbf{y}_2) \propto p_1(\mathbf{y}_1) \exp(\lambda \cdot \phi'(\mathbf{y}_1)) \; p_2(\mathbf{y}_2) \exp(-\lambda \cdot \phi'(\mathbf{y}_2)). \tag{7.6}$$

Evaluating Equation 7.6 with $\mathbf{y}_1 = \mathbf{y}_2 = \mathbf{y}$ we have $q(\mathbf{y})^2 = p_1(\mathbf{y})p_2(\mathbf{y})$ which gives us a closed form computation of $\mathbf{agree}(p_1, p_2) \propto \sqrt{p_1(\mathbf{y})p_2(\mathbf{y})}$. Substituting this solution into the problem of Proposition 7.1, and performing algebraic simplification yields the desired result. □

Replacing the minimum KL term in Equation 2.4 with a Bhattacharyya regularization term yields the objective

$$\min_{\theta} \mathcal{L}_1(\theta_1) + \mathcal{L}_2(\theta_2) + c\mathbf{E}_U[B(p_1(\theta_1), p_2(\theta_2))] \tag{7.7}$$

where $\mathcal{L}_i = \mathbf{E}[-\log(p_i(\mathbf{y}_i|\mathbf{x};\theta_i))] + \frac{1}{\sigma_i^2}||\theta_i||^2$ for $i = 1, 2$ are the standard regularized log likelihood losses of the models p_1 and p_2, $\mathbf{E}_U[B(p_1, p_2)]$ is the expected Bhattacharyya distance [Kailath, 1967] between the predictions of the two models on the unlabeled data, and c is a constant defining the relative weight of the unlabeled data relative to the labeled data.

Our regularizer extends to full agreement for undirected graphical models. In the case where p_1 and p_2 have the same structure, $q = \mathbf{agree}(p_1, p_2)$ will share this structure and the projection can be computed in closed form.

Proposition 7.2. *Suppose $p_i(\mathbf{Y}|\mathbf{X}), i \in \{1, 2\}$ factor as a set of clique potentials from a set of cliques $\mathcal{C}$:*

$$p_i(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z_i(\mathbf{X})} \prod_{c \in \mathcal{C}} \psi_i(\mathbf{X}, \mathbf{Y}_c)$$

then $q_i(\mathbf{Y})$ also factors as a product over clique potentials in $\mathcal{C}$, and can be computed in closed form modulo normalization as $q(\mathbf{Y}_1, \mathbf{Y}_2) = q_1(\mathbf{Y}_1)q_2(\mathbf{Y}_2)$ with

$$q_i(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z'(\mathbf{X})} \prod_{c \in \mathcal{C}} \sqrt{\psi_1(\mathbf{X}, \mathbf{Y}_c)\psi_2(\mathbf{X}, \mathbf{Y}_c)} \tag{7.8}$$

Proof. The proof is simple algebraic manipulation. We start with Equation 7.9, an application of Proposition 7.1.

$$\begin{aligned} q_i(\mathbf{Y})^2 &\propto p_1(\mathbf{Y}|\mathbf{X})p_2(\mathbf{Y}|\mathbf{X}) && (7.9)\\ &= Z_1^{-1}Z_2^{-1} \prod_c \psi_1(\mathbf{X}, \mathbf{Y}_c)\psi_2(\mathbf{X}, \mathbf{Y}_c) && (7.10)\\ &= \left(\frac{1}{Z'(\mathbf{X})} \prod_c \sqrt{\psi_1(\mathbf{X}, \mathbf{Y}_c)\psi_2(\mathbf{X}, \mathbf{Y}_c)} \right)^2 && (7.11) \end{aligned}$$

□

Note that Proposition 7.2 is not a special case of Proposition 2.2 because we have defined one constraint feature $\phi_{\mathbf{y}}$ for each possible labeling $\mathbf{y}$, and these do not decompose

according to $\mathcal{C}$. We could alternatively have proven that ensuring agreement on clique potentials is identical to ensuring agreement on labelings. In the case of log-linear Markov random fields, the clique potentials are stored in log space so computing q corresponds to averaging the values before computing normalization.

7.2 Partial Agreement and Hierarchical Labels

Our method extends naturally to partial-agreement scenarios. For example we can encourage two part-of-speech taggers with different tag sets to produce compatible parts of speech, such as noun in tag set one and singular-noun in tag set 2, as opposed to noun in tag set 1 and verb in tag set 2. In particular, suppose we have a mapping from both label sets into a common space where it makes sense to encourage agreement. For the part-of-speech tagging example, this could mean mapping all nouns from both tag sets into a single class, all verbs into another class and so on. In general suppose we have functions $g_1(\mathbf{y}_1)$ and $g_2(\mathbf{y}_2)$ that map variables for the two models onto the same space $\{\mathbf{z}\}$. Then, $p_i(\mathbf{y}_i)$ and g_i induce a distribution:

$$p_i(\mathbf{z}) = \sum_{\mathbf{y} \,:\, g_i(\mathbf{y})=\mathbf{z}} p_i(\mathbf{y}) \quad \text{and} \quad p_i(\mathbf{y}_i|\mathbf{z}) = p_i(\mathbf{y}_i)/p_i(\mathbf{z}).$$

We can encourage $p_1(\mathbf{z}) \approx p_2(\mathbf{z})$ by adding a feature for each label in the joint space:

$$\phi_{\mathbf{z}}(\mathbf{y}_i) = \begin{cases} 1 & \text{if } i = 1 \text{ and } g_1(\mathbf{y}_1) = \mathbf{z} \\ -1 & \text{if } i = 2 \text{ and } g_2(\mathbf{y}_2) = \mathbf{z} \\ 0 & \text{otherwise.} \end{cases} \tag{7.12}$$

In this case our objective becomes:

$$\min_{\theta} \mathcal{L}_1(\theta_1) + \mathcal{L}_2(\theta_2) + c\mathbf{E}_U[B(p_1(\mathbf{z}), p_2(\mathbf{z}))] \,. \tag{7.13}$$

In the special case where some labels are identical for the two models and others are incompatible, we have $g_1(\mathbf{z}_1)$ mapping the incompatible labels into one bin and the others into their own special bins. Proposition 7.3 along with the optimization algorithm described in Section 2.5 allows us to optimize this objective.

Proposition 7.3. *The Bhattacharyya distance* $-\log \sum_{\mathbf{z}} \sqrt{p_1(\mathbf{z})p_2(\mathbf{z})}$ *is* $\frac{1}{2}$ *the value of the convex optimization problem*

$$\begin{aligned} \min_{q} \quad & \mathbf{KL}(q(\mathbf{Y}_1, \mathbf{Y}_2)||p_1(\mathbf{Y}_1)p_2(\mathbf{Y}_2)) \\ \text{s. t.} \quad & \mathbf{E}_q(\phi) = 0, \end{aligned}$$

where the constraint features ϕ *are defined as in Equation 7.12. Furthermore, the minimizer decomposes as* $q(\mathbf{Y}_1, \mathbf{Y}_2) = q_1(\mathbf{Y}_1|\mathbf{z}_1)q_1(\mathbf{z}_1)q_2(\mathbf{Y}_2|\mathbf{z}_2)q_2(\mathbf{z}_2)$*, where* $q_1(\mathbf{z}_1) = q_2(\mathbf{z}_2) \propto \sqrt{p_1(\mathbf{z}_1)p_2(\mathbf{z}_2)}$ *and* $q_i(\mathbf{Y}_i|\mathbf{z}_i) = p_i(\mathbf{Y}_i|\mathbf{z}_i)$ $i \in \{1, 2\}$.

Note that the computation of $\mathbf{agree}(p_1, p_2)$ is still in closed form if our models are unstructured.

Unfortunately, if we collapse some labels for structured models, $p(\mathbf{Y})$ might not have the same Markov properties as $p(\mathbf{z})$. For example, consider the case where p is a distribution over three states (1,2,3) that assigns probability 1 to the sequence (1,2,3,1,2,3,...) and probability zero to other sequences. This is a first-order Markov chain. If the mapping is $1 \mapsto 1$ and $2, 3 \mapsto 0$ then $p(\mathbf{y})$ assigns probability 1 to (1,0,0,1,0,0,...), which cannot be represented as a first-order Markov chain. Essentially, the original chain relied on being able to distinguish between the allowable transition (2,3) and the disallowed transition (3,2). When we collapse the states, both of these transitions map to (0,0) and cannot be distinguished. Consequently, the closed form solution given in Proposition 7.3 is not usable. Potentially, we could compute some approximation to $p(\mathbf{y})$ and from that compute an approximation to q. Instead, we re-formulate our constraints to require only that the marginals of each clique in p_1 and p_2 match each other rather than requiring the joint to have the same probability:

$$\phi_{c,\mathbf{z}_c}(\mathbf{Y}_1, \mathbf{Y}_2) = \begin{cases} 1 & \text{if } i = 1 \text{ and } g_1(\mathbf{y}_1)_c = \mathbf{z}_c \\ -1 & \text{if } i = 2 \text{ and } g_2(\mathbf{y}_2)_c = \mathbf{z}_c \\ 0 & \text{otherwise.} \end{cases} \tag{7.14}$$

By Proposition 2.2, the features in Equation 7.14 lead to a q that respects the Markov properties of the original models.

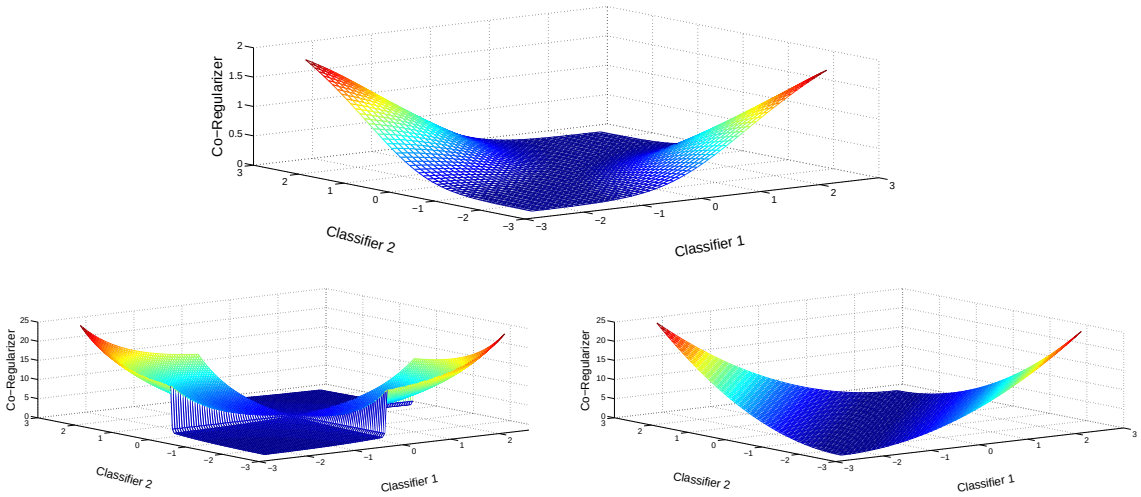


Figure 7.1: Different Loss Functions. **Top**: Bhattacharyya distance regularization. **Bottom left**: Exp-loss regularization. **Bottom right**: Least squares regularization.

7.3 Relation to Other Multi-View Learning

To avoid a long detour from PR, we describe here only CoBoosting [Collins and Singer, 1999] and two view Perceptron [Brefeld et al., 2005], the frameworks with which we empirically compare our method in the next section. Since these methods are based on different objective functions from ours it is worth examining where each one works best. Altun et al. [2003] compare log-loss and exp-loss for sequential problems. They find that the loss function does not have as great an effect on performance as the feature choice. However, they also note that exp-loss is expected to perform better for clean data, while log-loss is expected to perform better when there is label noise. The intuition behind this is in the rate of growth of the loss functions. Exp-loss grows exponentially with misclassification margin while log-loss grows linearly. Consequently when there is label noise, AdaBoost focuses more on modeling the noise. Since CoBoosting optimizes a co-regularized exp-loss while our work optimizes a co-regularized log-loss we expect to do better on problems where the labels are noisy.

To get more intuition about this, Figure 7.1 shows the co-regularization loss functions for our method, CoBoosting, and co-regularized least squares [Sindhwani et al., 2005]. For two underlying binary linear classifiers, $\hat{y}_1 = \text{sign}(w_1 \cdot x)$ and $\hat{y}_2 = \text{sign}(w_2 \cdot x)$, the hori-

zontal axes represent the values $\hat{y}_1$ and $\hat{y}_2$, while the vertical axis is the loss. If we consider the plane parallel to the page, we see how the different co-regularizers penalize the classifiers when they disagree and are equally confident in their decision. Restricted to this plane, all three co-regularizers grow at the same asymptotic rate as the loss functions for the individual models: Linearly for our work, exponentially for CoBoosting and quadratically for co-RLS. If we look at the area where the two models agree (the flat part of the CoBoosting graph) we see what the penalty is when the classifiers agree but have different confidence. In this case co-RLS is harshest since it penalizes differences in the dot product equally regardless of the absolute value of the dot product. Intuitively, this is a problem. If one model predicts 1 with confidence 0.5 and the other predicts -1 with confidence 0.5 they are disagreeing while if they both predict 1 with confidence 1000 and 1001 respectively, they are agreeing on the label and are very close in their confidence estimates. At the other extreme, CoBoosting imposes almost no penalty whenever the two classifiers agree, regardless of their confidence. The Bhattacharyya distance co-regularizer lies between these extremes, penalizing differences in confidence near the origin but is more lenient when the classifiers are both very confident and agree.

Finally, if we have labeled data from one domain but want to apply it to another domain we can use any of the co-training frameworks mentioned earlier, including our own, to perform domain transfer. For sentiment classification we will see that our method performs comparably with Structural Correspondence Learning [Blitzer et al., 2006], which is based on Alternating Structure Optimization [Ando and Zhang, 2005].

7.4 Experiments

Our first set of experiments is for transfer learning for sentiment classification. We use the data from Blitzer et al. [2007]. The two views are generated from a random split of the features. We compare our method to several supervised methods as well as CoBoosting [Collins and Singer, 1999], two view Perceptron [Brefeld et al., 2005] and structural correspondence learning [Blitzer et al., 2007]. Results are in Table 7.1. The column labeled "SCL" contains the best results from Blitzer et al. [2007], and is not directly comparable

Domains	MIRA	Boost	Perc	mx-ent	SCL	CoBoost	coPerc	PR
books→dvds	77.2	72.0	74	78.5	75.8	78.8	75.5	**79.8**
dvds→books	72.8	74.8	74.5	80.3	79.7	79.8	74.5	**81.3**
books→electr	70.8	70.3	73.3	72.5	75.9	**77.0**	69.3	75.5
electr→books	70.7	62.5	73	72.8	**75.4**	71.0	67.5	74.3
books→kitchn	74.5	76.3	73.5	77.8	78.9	78.0	76.5	**81.0**
kitchn→books	70.9	66.5	67.3	70.3	68.6	69.8	66	**72.8**
dvds→electr	73.0	73.2	73.5	75.5	74.1	75.3	71.2	**76.5**
electr→dvds	70.6	66.3	64.8	69.3	**76.2**	73.5	63.3	73.0
dvds→kitchn	74.0	75.5	78.3	80.5	81.4	79.0	78.25	**82.8**
kitchn→dvds	72.7	61.8	64	69.5	**76.9**	70.1	60.5	72.8
electr→kitchn	84.0	73.2	81	**86.5**	85.9	85.0	83.3	85.8
kitchn→electr	82.7	66.3	81	82.8	**86.8**	83.0	80.5	85.5
Avg. improvement	-1.87	-6.47	-3.18	N/A	1.61	0.33	-4.16	**2.07**
Std. deviation	3.04	4.73	2.21	N/A	3.31	2.03	2.12	**1.27**

Table 7.1: Performance of several methods on a sentiment classification transfer learning task. Reviews of objects of one type are used to train a classifier for reviews of objects of another type. The abbreviations in the column names are as follows. Boost: AdaBoost algorithm, Perc: Perceptron, mx-ent: maximum entropy, SCL: structural correspondence learning, CoBoost: CoBoosting, coPerc: two view Perceptron, PR: this work. The best accuracy is shown in bold for each task. The last two rows of the table show the average improvement over maximum entropy (the best performing supervised method), and also the standard deviation of the improvement.

with the other methods since it uses some extra knowledge about the transfer task to choose auxiliary problems. For all the two-view methods we weigh the total labeled data equally with the total unlabeled data. We regularize the maximum entropy classifiers with a unit variance Gaussian prior. Out of the 12 transfer learning tasks, our method performs best in 6 cases, SCL in 4, while CoBoosting performs best only once. Two view Perceptron never outperforms all other methods. One important reason for the success of our method is the relative strength of the maximum entropy classifier relative to the other supervised methods for this particular task. We expect that CoBoosting will perform better than our method in situations where Boosting significantly out-performs maximum entropy.

The next set of our experiments are on named entity disambiguation. Given a set of already segmented named entities, we want to predict what type of named entity each one is. We use the training data from the 2003 CoNLL shared task [Sang and Meulder, 2003]. The two views comprise content versus context features. The content features are words,

Data size	max-ent	$\mathbf{agree}_0$	PR	RRE
500	74.0	74.4	76.4	9.2%
1000	80.0	80.0	81.7	8.5%
2000	83.4	83.4	84.8	8.4%

Table 7.2: Named entity disambiguation. Prior variance and c chosen by cross validation. $\mathbf{agree}_0$ refers to performance of two view model before first iteration of EM. RRE is reduction in error relative to error of MaxEnt model.

POS tags and character n-grams of length 3 for all tokens in the named entity, while context features the same but for three words before and after the named entity. We used 2000 examples as test data and roughly 30,000 as unlabeled (train) data. Table 7.2 shows the results for different amounts of labeled train data. For this data, we choose the variance of the Gaussian prior as well as the relative weighting of the labeled and unlabeled data by cross validation on the train set. In order to test whether the advantage our method gets is from the joint objective or from the use of $\mathbf{agree}(p_1, p_2)$, which is an instance of logarithmic opinion pools, we also report the performance of using $\mathbf{agree}(p_1, p_2)$ when the two views p_1 and p_2 have been trained only on the labeled data. In the column labeled "$\mathbf{agree}_0$" we see that for this dataset the benefit of our method comes from the joint objective function rather than from the use of logarithmic opinion pools. Note that we use different data than the ones in Collins and Singer [1999], so our raw performance numbers are not comparable to theirs.

In order to investigate the applicability of our method to structured learning we apply it to the shallow parsing task of noun phrase chunking. We our experiments are on the English training portion of the CoNLL 2000 shared task [Sang and Buchholz, 2000a]. We select 500 sentences as test data and varying amounts of data for training; the remainder was used as unlabeled (train) data. We use content and context views, where the content view is the current word and POS tag while the context view is the previous and next words and POS tags. We regularize the CRFs with a variance 10 Gaussian prior and weigh the unlabeled data so that it has the same total weight as the labeled data. The variance value was chosen based on preliminary experiments with the data. Table 7.3 shows the F-1 scores of the different models. We compare our method to a monolithic CRF as well as averaged Perceptron the two view Perceptron of Brefeld et al. [2005] with averaging. The Perceptron models

size	CRF	SAR(RRE)	Perc	coPerc
10	73.2	**78.2** (19%)	69.4	71.2
20	79.4	**84.2** (23%)	74.4	76.8
50	86.3	**86.9** (4%)	80.1	84.1
100	88.5	**88.9** (3%)	86.1	88.1
200	89.6	89.6 (0%)	89.3	**89.7**
500	**91.3**	90.6 (-8%)	90.8	90.9
1000	91.6	91.1 (-6%)	91.5	**91.8**

Table 7.3: F-1 scores for noun phrase chunking with context/content views. Test data comprises 500 sentences, with 8436 sentences divided among labeled and unlabeled train data. The best score is shown in bold for each train data size.

were trained for 20 iterations. Preliminary experiments show that performance on held out data does not change after 10 iterations so we believe the models have converged. Both two view semi-supervised methods show gains over the corresponding fully-supervised method for 10-100 sentences of training data, but do not improve further as the amount of labeled data increases. The method presented in this dissertation out-performs two view Perceptron when the amount of labeled data is very small, probably because regularized CRFs perform better than Perceptron for small amounts of data. As the number of training sentences increases, two view Perceptron performs as well as our method, but at this point it has little or no improvement over the fully-supervised Perceptron.

Chapter 8

Cross lingual projection

This chapter is based on Ganchev et al. [2009a,b, 2010]. For English and a handful of other languages, there are large, well-annotated corpora with a variety of linguistic information ranging from named entity to discourse structure. Unfortunately, for the vast majority of languages very few linguistic resources are available. This situation is likely to persist because of the expense of creating annotated corpora that require linguistic expertise [Abeillé, 2003]. On the other hand, parallel corpora between many resource-poor languages and resource-rich languages are ample, motivating recent interest in transferring linguistic resources from one language to another via parallel text.

Dependency grammars are one such resource. They are useful for language modeling, textual entailment and machine translation [Haghighi et al., 2005, Chelba et al., 1997, Quirk et al., 2005, Shen et al., 2008], to name a few tasks. Dependency grammars are arguably more robust to transfer than constituent grammars, since syntactic relations between aligned words of parallel sentences are better conserved in translation than phrase structure [Fox, 2002, Hwa et al., 2005]. The two main challenges to accurate training and evaluation from aligned bitext are: (1) errors in word alignments and source language parses, (2) unaligned words due to non-literal or distant translation.

Hwa et al. [2005] proposed to learn generative dependency grammars using Collins' parser [Collins, 1999] by constructing full target parses via projected dependencies. To address challenge (1), they introduced on the order of one to two dozen language-specific transformation rules. To address challenge (2), they used a set of tree-completion rules.

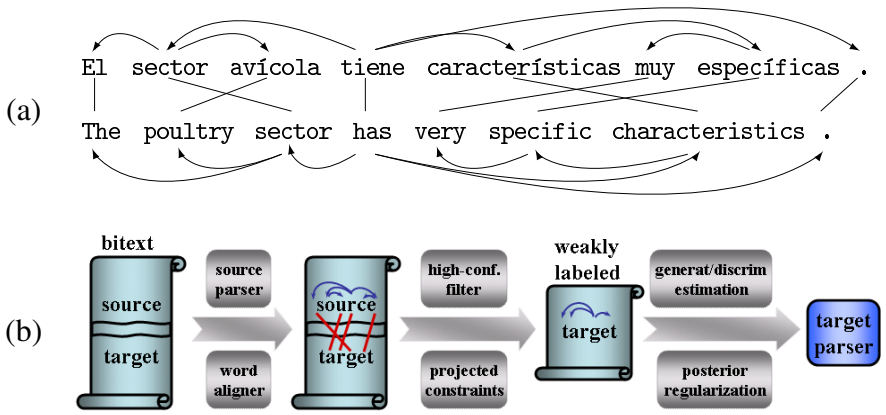


Figure 8.1: (a) An example word-aligned sentence pair with perfectly projected dependencies. (All dependency edges are conserved in this case.) (b) Overview of our grammar induction approach via bitext: the source (English) is parsed and word-aligned with target; after filtering, projected dependencies define constraints over target parse tree space, providing weak supervision for learning a target grammar.

We present here an alternative approach to dependency grammar transfer. Our approach uses a single, intuitive PR constraint to guide grammar learning. With this constraint, we avoid the need for complex tree completion rules and many language-specific rules, yet still achieve acceptable parsing accuracy.

It should be noted that while our source of supervision, a bitext, is the same as that of Hwa et al. [2005], our learning method is more closely related to that of Druck et al. [2009]. They use the GE framework to train a dependency parser. Their source of supervision comes in the form of corpus-wide expected values of linguistic rules provided by a linguistic informant.

In what follows, X will indicate parallel part-of-speech tagged sentences in a bitext corpus, along with a dependency parse of the source language. Y will indicate the dependency parses for the target language sentences.

8.1 Approach

Figure 8.1(a) shows an aligned sentence pair example where dependencies are perfectly "conserved" across the alignment. An edge from English parent p to child c is called conserved if word p aligns to word p' in the second language, c aligns to c' in the second

language, and p' is the parent of c'. Note that we are not restricting ourselves to one-to-one alignments here; p, c, p', and c' can all also align to other words. Unfortunately the sentence in Figure 8.1(a) is highly unusual in its amount of dependency conservation, so we need to do more than directly transfer conserved edges to get good parsing accuracy.

The key to our approach is a single PR constraint, which ensures that the expected proportion of conserved edges in a sentence pair is at least η (the exact proportion we used was 0.9, which was determined using unlabeled data as described in the experiments section). Specifically, let $C_{\mathbf{x}}$ be the set of directed edges projected from English for a given sentence $\mathbf{x}$. Then given a parse $\mathbf{y}$, the proportion of conserved edges is $\phi(\mathbf{x}, \mathbf{y}) = \frac{1}{|C_{\mathbf{x}}|} \sum_{y \in \mathbf{y}} \mathbf{1}(y \in C_{\mathbf{x}})$ and the expected proportion of conserved edges under distribution $p(\mathbf{y} \mid \mathbf{x})$ is

$$\mathbf{E}_p[\phi(\mathbf{x}, \mathbf{y})] = \frac{1}{|C_{\mathbf{x}}|} \sum_{y \in C_{\mathbf{x}}} p(y \mid \mathbf{x}). \tag{8.1}$$

Consider how this constraint addresses errors in word alignment and source language parses, challenge (1) from above. First, note that we are constraining groups of edges rather than a single edge. For example, in some sentence pair we might find 10 edges that have both end points aligned and can be transferred. Rather than requiring our target language parse to contain each of the 10 edges, we require that the expected number of edges from this set is at least 10η. This gives the parser freedom to have some uncertainty about which edges to include, or alternatively to choose to exclude some of the transferred edges.

Our constraint does not address unaligned words due to non-literal or distant translation, challenge (2), as directly. Yet, we find that the constraint sufficiently limits the distribution over possible parses of unaligned words, such that the parser still makes reasonable choices for them. It is also worth noting that if we wished to more directly address challenge (2), we could add additional constraint features to the PR framework. For example, it seems intuitive that unaligned words might tend to be leaves (e.g. articles that are dropped in some languages but not in others). Thus, one constraint we could enforce would be to restrict the number of children of unaligned words to fall below some threshold.

For both models, we compute the projection onto the constraint set using a simple line search. This is possible since there is only one constraint per sentence and the constraints

do not interact.

At a high level our approach is illustrated in Figure 8.1(b). A parallel corpus is word-level aligned using the method described in Chapter 6, where we train an alignment model via PR using symmetry constraints. The source (English) is parsed using a dependency parser [McDonald et al., 2005]. Then, the filtering stage eliminates low-confidence alignments such as noun-to-verb alignments, restricts the training set to exclude possible sentence fragments, and follows the method of Klein and Manning [2004] in stripping out punctuation. We then learn a probabilistic parsing model using PR. In our experiments we evaluate the learned models on dependency treebanks [Nivre et al., 2007].

8.2 Parsing Models

We explored two parsing models: a generative model used by several authors for unsupervised induction and a discriminative model previously used for fully supervised training.

The discriminative parser is based on the edge-factored model and features of the MSTParser [McDonald et al., 2005]. The parsing model defines a conditional distribution $p_\theta(\mathbf{y} \mid \mathbf{x})$ over each projective parse tree $\mathbf{y}$ for a particular sentence $\mathbf{x}$, parameterized by a vector θ. The probability of any particular parse is

$$p_\theta(\mathbf{y} \mid \mathbf{x}) \propto \prod_{y \in \mathbf{y}} e^{\theta \cdot \mathbf{f}(y, \mathbf{x})}, \tag{8.2}$$

where y is a directed edge contained in the parse tree $\mathbf{y}$ and $\mathbf{f}$ is a feature function. In the fully supervised experiments we run for comparison, parameter estimation is performed by stochastic gradient ascent on the conditional likelihood function, similar to maximum entropy models or conditional random fields. One needs to be able to compute expectations of the features $\mathbf{f}(y, \mathbf{x})$ under the distribution $p_\theta(y \mid \mathbf{x})$. A version of the inside-outside algorithm [Lee and Choi, 1997] performs this computation. Viterbi decoding is done using Eisner's algorithm [Eisner, 1996].

We also used a generative model based on dependency model with valence [Klein and Manning, 2004]. Under this model, the probability of a particular parse $\mathbf{y}$ and a sentence

Basic Uni-gram Features
x_i-word, x_i-pos
x_i-word
x_i-pos
x_j-word, x_j-pos
x_j-word
x_j-pos

Basic Bi-gram Features
x_i-word, x_i-pos, x_j-word, x_j-pos
x_i-pos, x_j-word, x_j-pos
x_i-word, x_j-word, x_j-pos
x_i-word, x_i-pos, x_j-pos
x_i-word, x_i-pos, x_j-word
x_i-word, x_j-word
x_i-pos, x_j-pos

In Between POS Features
x_i-pos, b-pos, x_j-pos

Surrounding Word POS Features
x_i-pos, x_i-pos+1, x_j-pos-1, x_j-pos
x_i-pos-1, x_i-pos, x_j-pos-1, x_j-pos
x_i-pos, x_i-pos+1, x_j-pos, x_j-pos+1
x_i-pos-1, x_i-pos, x_j-pos, x_j-pos+1

Table 8.1: Features used by the MSTParser. For each edge (i, j), x_i-word is the parent word and x_j-word is the child word, analogously for POS tags. The +1 and -1 denote preceeding and following tokens in the sentence, while b denotes tokens between x_i and x_j.

with part-of-speech tags $\mathbf{x}$ is given by

$$p_\theta(\mathbf{y}, \mathbf{x}) = p_{\text{root}}(r(\mathbf{x})) \cdot \Big(\prod_{y \in \mathbf{y}} p_{\neg\text{stop}}(y_p, y_d, v_y)\, p_{\text{child}}(y_p, y_d, y_c) \Big) \cdot \Big(\prod_{x \in \mathbf{x}} p_{\text{stop}}(x, \text{left}, v_l)\, p_{\text{stop}}(x, \text{right}, v_r) \Big) \tag{8.3}$$

where $r(\mathbf{x})$ is the part-of-speech tag of the root of the parse tree $\mathbf{y}$, y is an edge from parent y_p to child y_c in direction y_d, either left or right, and v_y indicates valency—false if y_p has no other children further from it in direction y_d than y_c, true otherwise. The valencies v_r/v_l are marked as true if x has any children on the left/right in $\mathbf{y}$, false otherwise.

We regularize the models by parameter prior $-\log p(\theta) = R(\theta)$, where $p(\theta)$ is Gaussian for the discriminative model and Dirichlet for the generative.

8.3 Experiments

We evaluate our approach by transferring from an English parser trained on the Penn treebank to Bulgarian and Spanish. We evaluate our results on the Bulgarian and Spanish corpora from the CoNLL X shared task. The Bulgarian experiments transfer a parser from English to Bulgarian, using the OpenSubtitles corpus [Tiedemann, 2007]. The Spanish experiments transfer from English to Spanish using the Spanish portion of the Europarl corpus [Koehn, 2005]. For both corpora, we performed word alignments with the open source PostCAT [Graça et al., 2009c] toolkit. We used the Tokyo tagger [Tsuruoka and

Tsujii, 2005] to POS tag the English tokens, and generated parses using the first-order model of McDonald et al. [2005] with projective decoding, trained on sections 2-21 of the Penn treebank with dependencies extracted using the head rules of Yamada and Matsumoto [2003]. For Bulgarian we trained the Stanford POS tagger [Toutanova et al., 2003] on the Bulgtreebank corpus from CoNLL X. The Spanish Europarl data was POS tagged with the FreeLing language analyzer [Atserias et al., 2006]. The discriminative model used the same features as MSTParser, summarized in Table 8.1. Our model uses constraints of the form: the expected proportion of conserved edges in a sentence pair is at least $\eta = 90\%$.[1]

In order to better evaluate our method, we construct a baseline inspired by Hwa et al. [2005]. The baseline creates a full parse tree from the incomplete and possibly conflicting transferred edges using a simple random process. We start with no edges and try to add edges one at a time verifying at each step that it is possible to complete the tree. We first try to add the transferred edges in random order, then for each orphan node we try all possible parents (both in random order). We then use this full labeling as supervision for a parser. Note that this baseline is very similar to the first iteration of our model, since for a large corpus the different random choices made in different sentences tend to smooth each other out. We also tried to create rules for the adoption of orphans, but the simple rules we tried added bias and performed worse than the baseline we report.

8.4 Results

Models are evaluated based on attachment accuracy—the fraction of words assigned the correct parent. Figure 8.2 shows that models generally improve with more transfer-type data. It also shows our method consistently outperforms the baseline. Note that each point in these graphs is based on a single random subsample of the data, which leads to some non-monotonicity in the left-half of some of the curves. The exact accuracy numbers for the 10k training sentences point of Figure 8.2 are given in Table 8.2. Link-left baselines for these corpora are much lower: 33.8% and 27.9% for Bulgarian and Spanish respectively.

[1] We chose η in the following way: We split the unlabeled parallel text into two portions. We trained models with different η on one portion and ran it on the other portion. We chose the model with the highest fraction of conserved constraints on the second portion.

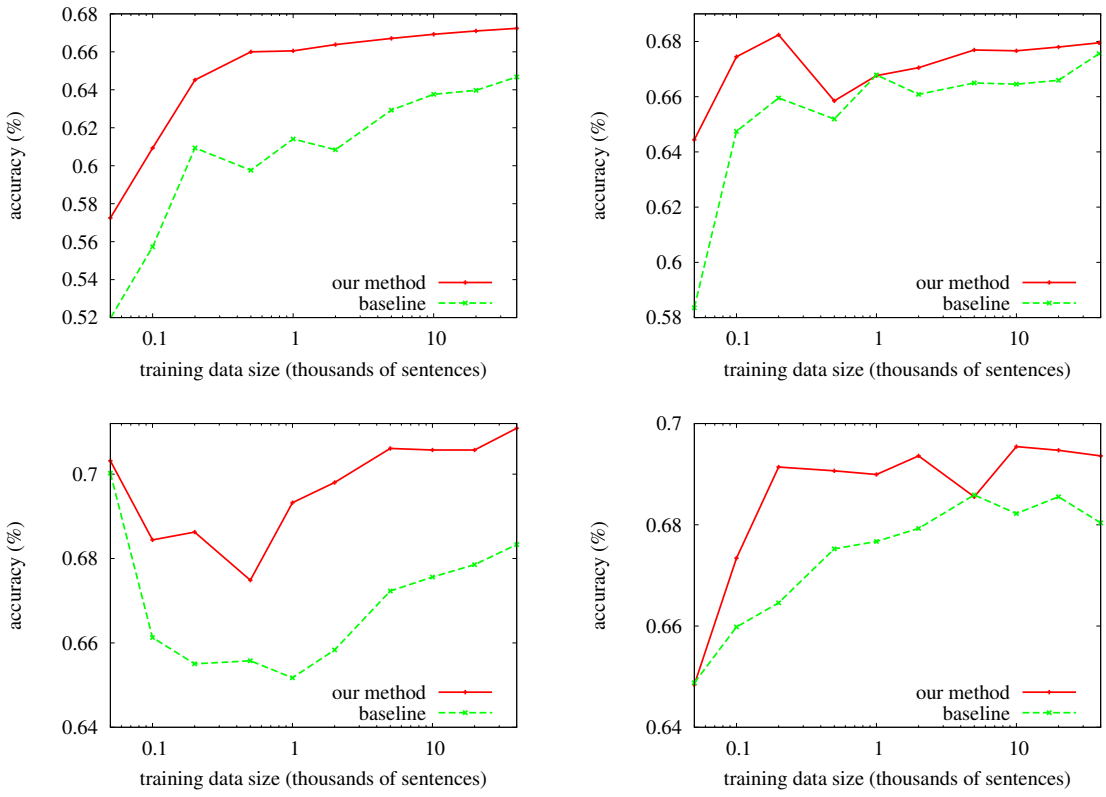


Figure 8.2: Learning curves. Each graph compares transferring a single tree of edges (baseline) and transferring all possible projected edges (our method). The models were trained on sentences of length up to 20 and tested on CoNLL train sentences of length up to 10. Punctuation was stripped at train time. **Top**: Bulgarian. **Bottom**: Spanish. **Left**: Discriminative model. **Right**: Generative model. The non-monotonicity of the (such as bottom left) is because each point is based on a single random sample of sentences. This random selection can greatly affect performance when the number of sentences is small.

	Discriminative		Generative	
	Bulgarian	Spanish	Bulgarian	Spanish
Baseline	63.8	67.6	66.5	68.2
Post.Reg.	**66.9**	**70.6**	**67.8**	**69.5**

Table 8.2: Accuracy values at the 10k training sentences point of Figure 8.2.

8.5 Generative Parser

The generative model we use is a state of the art model for unsupervised parsing. Before evaluating, we smooth the resulting models by adding e^{-10} to each learned parameter,

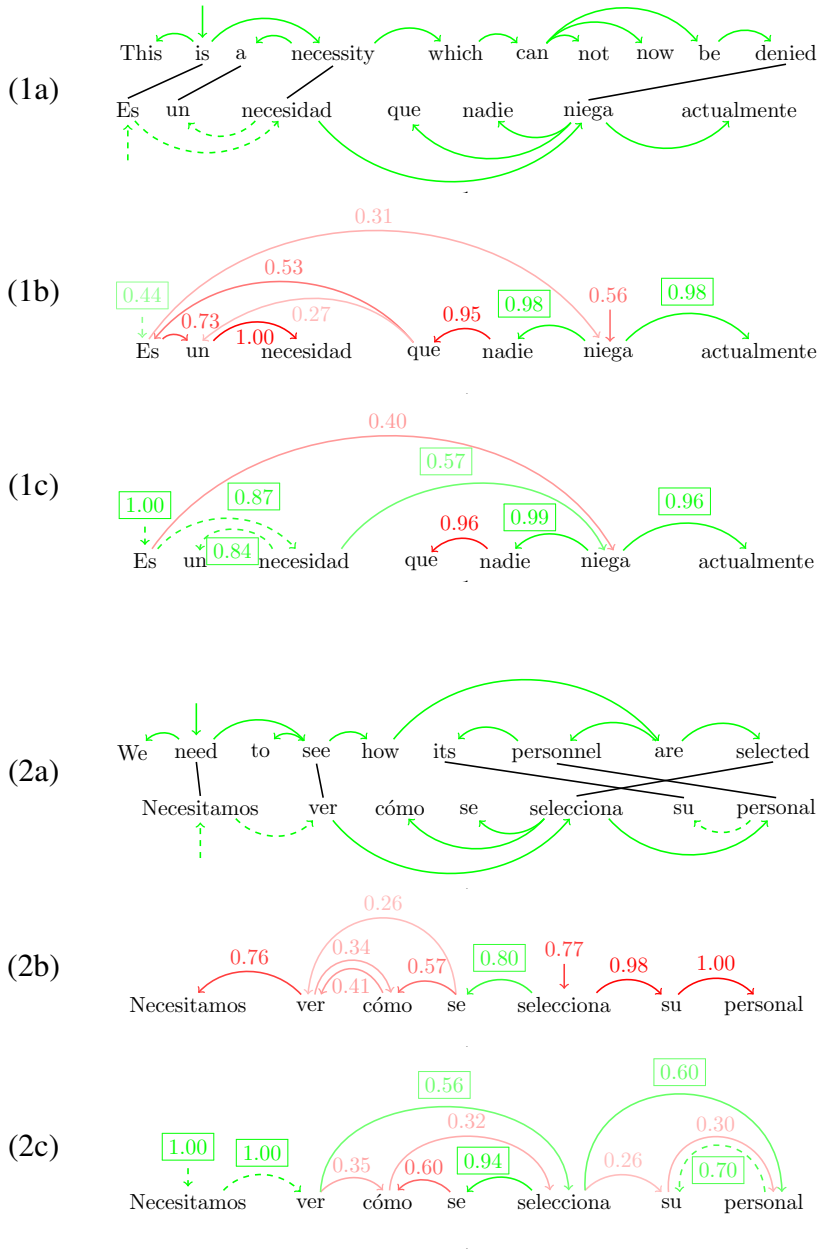


Figure 8.3: Posteriors of two Spanish sentences from Europarl. The number on each edge indicates the edge's posterior probability. Edges with probability less than 0.25 are not shown. Darker (more saturated) edges are higher probability. Green (with boxed number) indicates a correct edge, red (no box) an incorrect. Dotted edges are conserved. **(a)** The gold source and target parses and their alignment. **(b)** Unsupervised model initialized as per Klein and Manning [2004] and trained for 100 EM iterations. **(c)** PR projection applied to the posteriors of the middle figure, forcing $\mathbf{E}_p[|\mathbf{y} \cap C_\mathbf{x}|] \geq |C_\mathbf{x}| * \eta = 3 * 0.9$, where $C_\mathbf{x}$ is the set of transferred edges: {root→Necesitamos, Necesitamos→ver and personal→su}.

merely to remove the chance of zero probabilities for unseen events. (We did not bother to tune this value at all as it makes very little difference for final parses.) Unfortunately, we found generative model performance was disappointing in the unsupervised setting. Using the initialization procedure from Klein and Manning [2004], the maximum unsupervised accuracy it achieves is 55.4% for Bulgarian and 41.7% for Spanish, and these results are not stable. Changing the initialization parameters or training sample drastically affects the results, even for samples with several thousand sentences. But when we use the transferred information to constrain the learning, EM stabilizes and achieves much better performance, also beating the Hwa et al. [2005]-inspired baseline. With the transferred information, even setting all parameters equal at the outset does not prevent the model from learning the dependency structure of the aligned language. Figure 8.3 shows an example of how PR projection helps better estimate posteriors of two example sentences.

8.6 Discriminative Parser

Our discriminative parser is described in Section 8.2, and its features are illustrated in Table 8.1. Training was performed using an online version of the constrained EM algorithm described in Section 2.7. The online version proceeds as follows: for each sentence we use the current model parameters to compute the model's distribution over parse trees for the current sentence $p_\theta(\mathbf{y}|\mathbf{x})$. We then perform a min KL-projection that sentence to get $q(\mathbf{y})$. Because we have only one constraint, this only requires a line-search to find the optimal λ^*. We then perform a stochastic gradient step in the direction of $q(\mathbf{y})$ and then move on to the next sentence. We performed 100 such stochastic passes through the unlabeled constrained corpus. In all our experiments we used a Gaussian prior variance of 100.

The transfer system performs better than the unsupervised generative model and the baseline model for both Bulgarian and Spanish. We observed another desirable property of the discriminative model: While the generative model can get confused and perform poorly when the training data contains very long sentences, the discriminative parser does not appear to have this drawback. In fact we observed that as the maximum training sentence length increased, the parsing performance also improved.

Chapter 9

Enforcing sparsity structure for POS induction

This chapter is based on Graça et al. [2009a], Ganchev et al. [2009b, 2010]. Many important NLP tasks (e.g. tagging, parsing, named-entity recognition) involve word classification. Often, we know a priori that a word type might belong to a small set of classes (where the class of a specific instance depends on its context) and should never belong to any of the many possible classes outside this small set. The part-of-speech tagging task, described in the running example, is one instance of this phenomenon. For example, consider the word type "run". It might belong to the verb class in some instances and the noun class in others, but it will never be an adjective, adverb, conjunction, determiner, etc. Learning algorithms typically assume that each word type can be associated with any existing tag, even though in reality each word type is only ever associated with a few tags.

Unsupervised induction of this latent structure is normally performed using the EM algorithm, but it has exhibited disappointing performance in previous work. One well-known reason for this is that EM tends to allow each word to be generated by most POS tags some of the time. In reality, we would like most words to have a small number of possible POS tags. Previous work has attempted to solve this problem by applying the Bayesian approach, using a prior to encourage sparsity in the model *parameters* [Gao and Johnson, 2008, Johnson, 2007, Goldwater and Griffiths, 2007]. However, this approach has the drawback of enforcing sparsity in the wrong direction; sparsity at the parameter

level encodes a preference that each POS tag should generate only a few words, instead of encoding that each word should generate only a few POS tags. Here we explore the problem of biasing unsupervised models to favor the correct sparsity by encouraging the model to achieve *posterior* sparsity on unlabeled training data.

9.0.1 ℓ_1/ℓ_∞ Regularization for POS tagging

We focus on the slack-penalized formulation of Section 2.3 for this task. We choose the PR constraint to encourage each word to be associated with only a few parts of speech. Let the constraint feature $\phi_{wti}(\mathbf{X}, \mathbf{Y})$ have value 1 whenever the i^{th} occurrence of word w has part-of-speech tag t, and value 0 otherwise. For every word w, we would like there to be only a few POS tags t such that there are occurrences i where t has nonzero probability. This can be achieved if it "costs" a lot the first time an occurrence of a word takes a particular tag, but afterwards future occurrences of the word can receive that same tag for free. More precisely, for each word type w, we would like the sum (ℓ_1 norm), over tags t, of the maximum (ℓ_∞ norm), over all occurrences w_i of w, of $p(w_i \mid t)$, to be small; we want $\sum_{t,w} \max_i p(w_i \mid t)$ to be small. Note that in contrast to previous applications, these constraints are corpus-wide instead of instance-specific. For notational simplicity we will write $\phi_{wti}(\mathbf{X}, \mathbf{Y}) = \phi_{wti}(\mathbf{Y})$, since any dependence on $\mathbf{X}$ is captured in the subscripts of ϕ_{wti}.

Formally, this objective is an example of the slack-penalized formulation (as in Equation 2.5), but for simplicity we will use the notation:

$$\min_{q, c_{wt}} \ \mathbf{KL}(q||p_\theta) + \sigma \sum_{wt} c_{wt} \quad \text{s.t.} \quad \mathbf{E}_q[\phi_{wti}] \leq c_{wt}. \tag{9.1}$$

Mapping this notation to the original equation we have: $\mathbf{b} = 0$ and regularization strength σ. The constraints on the features ϕ_{wti} and the summation over c_{wt} together encode the ℓ_1/ℓ_∞ norm. The variables c_{wt} represent the ℓ_∞ norm of ϕ_{wti}, $c_{wt} = ||\phi_{wti}||_{\ell_\infty}$, while the summation is the ℓ_1 norm of c_{wt}. The dual of this objective has a very simple form:

$$\max_{\lambda \geq 0} \ -\log \left(\sum_y p_\theta(\mathbf{Y}) \exp(-\boldsymbol{\lambda} \cdot \phi(\mathbf{Y})) \right) \quad \text{s.t.} \quad \sum_i \lambda_{wti} \leq \sigma, \tag{9.2}$$

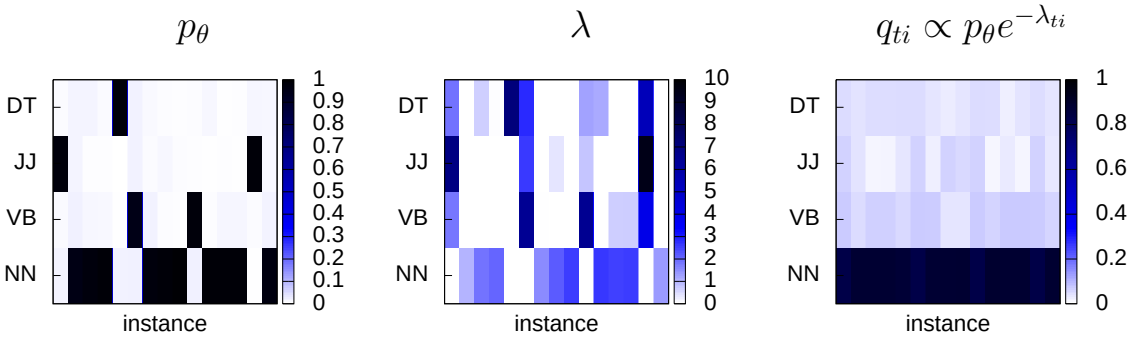


Figure 9.1: An illustration of ℓ_1/ℓ_∞ regularization. Left panel: initial tag distributions (columns) for 15 instances of a word. Middle panel: optimal regularization parameters λ, each row sums to $\sigma = 20$. Right panel: q concentrates the posteriors for all instances on the NN tag, reducing the ℓ_1/ℓ_∞ norm from just under 4 to a little over 1.

where $\mathbf{Y}$ ranges over assignments to the hidden tag variables for all of the occurrences in the training data, $\phi(\mathbf{Y})$ is the vector of ϕ_{wti} constraint feature values for assignment $\mathbf{Y}$, $\boldsymbol{\lambda}$ is the vector of dual parameters λ_{wti}, and the primal parameters are $q(\mathbf{Y}) \propto p_\theta(\mathbf{Y}) \exp(-\boldsymbol{\lambda} \cdot \phi(\mathbf{Y}))$.

An advantage of using slack penalties in this case is that ℓ_1/ℓ_∞ as a slack constraint in the primal would lead to a non-differentiable dual penalty term, which somewhat complicates optimization. Using a slack penalty makes sparsity regularization a primal penalty, yielding dual simplex constraints, solvable efficiently via projected gradient, as described by Bertsekas [1999]. Note that the simplex constraints in Equation 9.2 can be interpreted as an ℓ_∞/ℓ_1 norm, which is the dual of the ℓ_1/ℓ_∞.

Figure 9.1 illustrates how the ℓ_1/ℓ_∞ norm operates on a toy example. For simplicity suppose we are only regularizing one word and our model p_θ is just a product distribution over 15 instances of the word. The left panel in Figure 9.1 shows the posteriors under p_θ. We would like to concentrate the posteriors on a small subset of rows. The center panel of the figure shows the $\boldsymbol{\lambda}$ values determined by Equation 9.2, and the right panel shows the projected distribution q, which concentrates most of the posterior on the bottom row. Note that we are not requiring the posteriors to be sparse, which would be equivalent to preferring that the distribution is peaked; rather, we want a word to concentrate its tag posterior on a few tags across all instances of the word. Indeed, most of the instances (columns) become

	Types	Tokens	Unk	Tags	ℓ_1/ℓ_∞
PTB17	23768	950028	2%	17	1.23
PT-Conll	11293	206678	8.5%	22	1.14
BulTree	12177	174160	10%	12	1.04

Table 9.1: Corpus statistics. All words with only one occurrence where replaced by the 'unk' token. The third column shows the percentage of tokens replaced. ℓ_1/ℓ_∞ is the value of the sparsity for a fully supervised HMM trained in all available data.

less peaked than in the original posterior to allow posterior mass to be redistributed away from the outlier tags. Since they are more numerous than the outliers, they moved less. This also justifies only regularizing relatively frequent events in our model.

9.1 Results

In this section we present an empirical comparison of first-order HMMs trained with three different methods: classic EM (EM), ℓ_1/ℓ_∞ PR (Sparse), and Bayesian estimation using a variational approximation described in Johnson [2007] and Gao and Johnson [2008] (VEM). Models are trained and tested on three different corpora: the Wall Street Journal portion of the Penn treebank [Marcus et al., 1993] using a reduced set of 17 tags [Smith et al., 2005] (PTB17); the Bosque subset of the Portuguese Floresta Sinta(c)tica Treebank [Afonso et al., 2002][1] used for the ConLL X shared task on dependency parsing (PT-CoNLL)[2]; and the Bulgarian BulTreeBank [Simov et al., 2002] (BulTree) with 12 coarse tags. All words that occurred only once were replaced by the token "unk". To measure model sparsity, we compute the average ℓ_1/ℓ_∞ norm over words occurring more than 10 times; the label 'L1LMax' denotes this measure in figures. Table 9.1 gives statistics for each corpus as well as the sparsity for a first-order HMM trained on the labeled data.

Following Gao and Johnson [2008], the parameters were initialized with a "pseudo E-step" as follows: we filled the expected count matrices with numbers $1 + X \times U(0, 1)$, where $U(0, 1)$ is a random number between 0 and 1 and X is a parameter. These matrices are then fed to the M-step; the resulting "random" transition and emission probabilities

[1] http://www.linguateca.pt/Floresta/

[2] http://nextens.uvt.nl/ conll/

are used for the first real E step. For VEM X was set to 0.0001 (almost uniform) since this showed a significant improvement in performance. On the other hand EM showed less sensitivity to initialization, and we used $X = 1$ which resulted in the best results. The models were trained for 200 iterations as longer runs did not significantly change the results. For VEM we tested 4 different prior combinations based on the results of Johnson [2007]; in later work Gao and Johnson [2008] considered a wider range of values but did not identify definitely better choices. Sparse was initialized with the parameters obtained by running EM for 30 iterations, followed by 170 iterations of the new training procedure. Predictions were obtained using posterior decoding since this consistently showed small improvements over Viterbi decoding.

We compare the models by measuring the mutual information between the distribution of hidden states and the distribution of the truth. Ideally, a perfect method would have mutual information equal to the entropy of both distributions. The farther the distribution that a method produces is from the truth the smaller the information gain is. We also evaluate the accuracy of the models using two established mappings between hidden states and POS tags: **(1-Many**) maps each hidden state to the tag with which it co-occurs the most; **1-1** [Haghighi and Klein, 2006] greedily picks a tag for each state under the constraint of never using the same tag twice. This results in an approximation of the optimal 1-1 mapping. If the numbers of hidden states and tags are not the same, some hidden states will be unassigned (and hence always wrong) or some tags not used. In all our experiments the number of hidden states is the same as the number of POS tags.

Figure 9.2 (Top Left) shows mutual information between the hidden state distribution of each method and the truth. The entropy of the true distribution are: BulTree 3.05, PT-CoNLL 3.49 and PTB17 3.22. Sparse is the method that achieves the biggest information gain across all corpora, and is not particularly sensitive to the strength of regularization used. Interestingly, VEM often has the smallest ℓ_1/ℓ_∞, even though mutual information is often worst than EM.

Figure 9.2 (Top Right) shows the different average values of the L1LMax statistics for each method across corpora. We see that both VEM and Sparse achieve values of ℓ_1/ℓ_∞ close to the gold standard, on the other hand EM as expected as bigger values which

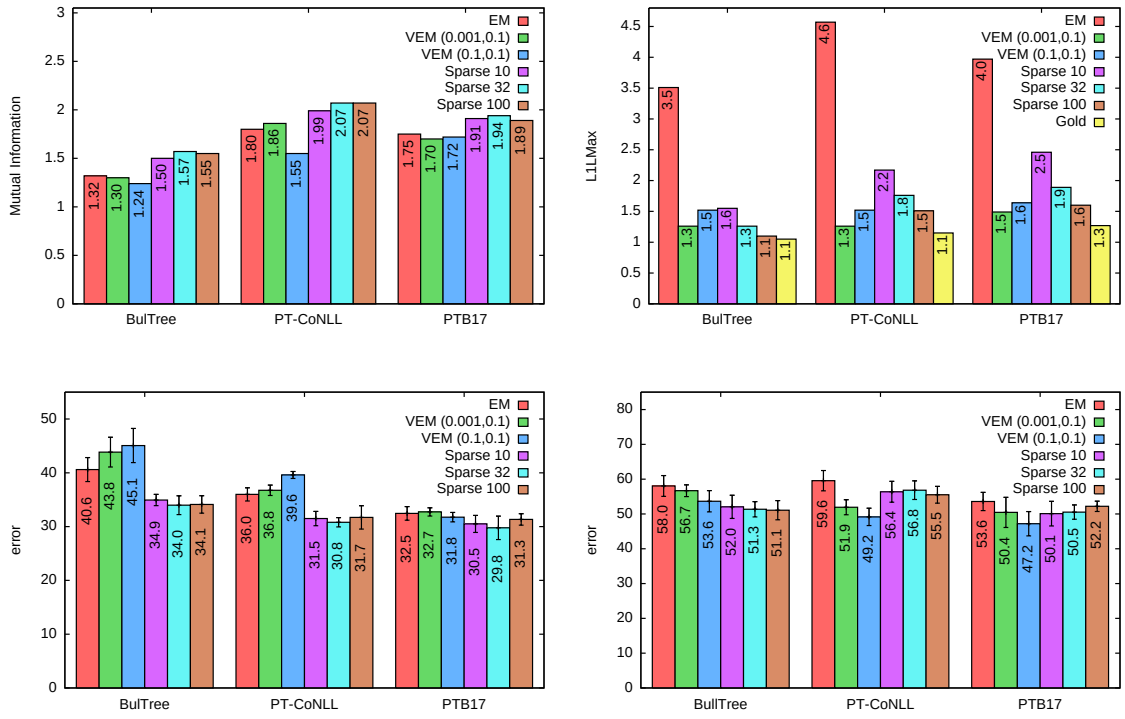


Figure 9.2: (Top Left) Mutual information in bits between gold tag distribution and hidden state distribution. The maximum is the entropy of the gold set (BulTree 3.05, PT-CoNLL 3.49 and PTB17 3.22), (Top Right) ℓ_1/ℓ_∞ value, and average (Bottom Left) 1-Many error, (Bottom Right) 1-1 error over 10 different runs (same seeds used for each model) for 200 iterations. Error bars are standard deviation for the 10 runs. All models are first order HMMs: EM trained using expectation maximization, VEM trained using variational EM using 0.1 state to state prior and (01,0.0001) observation prior; Sparse trained using PR with constraint strength $\sigma = 10, 32, 100$.

confirms the intuition that EM allows each word to be generated by most of the possible POS tags.

Figure 9.2 (Bottom Left) shows errors for all methods on the different corpora after 10 random initializations using the 1-Many mapping. For both VEM and Sparse we pick parameter settings resulting in the best average performance. A first conclusion is that using the ℓ_1/ℓ_∞ constraint consistently and significantly improves the results when compared with the other two methods.

Figure 9.2 (Bottom Right) shows the same errors for the 1-1 mapping. In this case Sparse still beats the EM but does not always outperform VEM. One reason for this behavior is that this metric is very sensitive to the number of word types associated with each

hidden state. VEM tends to encourage some large hidden states with many word types, which is preferable using the 1-1 mapping for large word categories such as nouns. On the other hand Sparse tends to spread the nouns over 4 different hidden states. This difference is particularly pronounced for the condensed tag sets (PTB17, PT-CoNLL) where different kinds of nouns are joined into one large tag. Also this difference is bigger for VEM when the observation prior is set to bigger values (0.1), leading at the same time to worse results in 1-Many mapping.

Chapter 10

Enforcing sparsity structure for Grammar induction

In this chapter, we investigate unsupervised learning methods for dependency parsing models that impose sparsity biases on the types of dependencies. We assume a corpus annotated with part-of-speech (POS) tags, where the task is to induce a dependency model from the tag sequences for corpus sentences. In this setting, the *type* of a dependency is defined as a simple pair: tag of the dependent (also known as the child), and tag of the head (also known as the parent) for that dependent. Given that POS tags are typically designed to convey information about grammatical relations, it is reasonable to assume that only some of the possible dependency types would be realized for any given language. For instance, it is ungrammatical for cardinal numbers to dominate verbs adjectives to dominate adverbs, and determiners to dominate almost any part of speech. In other words, the realized dependency types should be a sparse subset of all the possible types.

Previous work in unsupervised grammar induction has tried to achieve sparsity through priors on model parameters. For instance, Liang et al. [2007], Finkel et al. [2007] and Johnson et al. [2007] experimented with hierarchical Dirichlet process priors and Headden III et al. [2009] proposed a discounting Dirichlet prior. Such priors on parameters encourage a standard generative dependency parsing model (see Section 10.1) to limit the number of dependent types for each head type. Although not focused on sparsity, several other studies use soft parameter sharing to constrain the capacity of the model and hence couple differ-

ent types of dependencies. To this end, Cohen et al. [2008] and Cohen and Smith [2009] investigated a (shared) logistic normal prior, and Headden III et al. [2009] used a backoff scheme.

Our experiments (see Section 10.4) show that the more effective sparsity pattern is one that limits the total number of unique head-dependent tag pairs. Unlike sparsity-inducing parameter priors, this kind of sparsity bias does not induce competition between dependent types for each head type. We can achieve the desired bias with a sparsity constraint on model posteriors, similar to the constraints in Chapter 9.

As in Chapter 9, Specifically, to implement PR we augment the maximum likelihood objective of a generative model with a term that penalizes head tag-dependent tag distributions that are too permissive. In the case of parse trees, we consider two choices for the form of the penalty, and show experimentally that the following penalty works especially well: the model pays for the first time it selects a word with tag c as a dependent of a head with tag p; after that, choosing a head with that p for any other occurrence of c is free.

The model and all the code required to reproduce the experiments is available online at `http://code.google.com/p/pr-toolkit`.

10.1 Parsing Model

The models we consider are based on Klein and Manning [2004]'s dependency model with valence (DMV), introduced in Chapter 8. We also investigate extensions to the DMV borrowed from McClosky [2008] and Headden III et al. [2009]. These extensions are not crucial to our experimental success with posterior regularization, but we choose to explore them for better comparison with previous work. As will be discussed in the experiments section, both for the basic and for the extended models accuracy can be increased by applying posterior regularization. Section 10.1.1 describes the basic model and Section 10.1.2 describes the extensions we implemented.

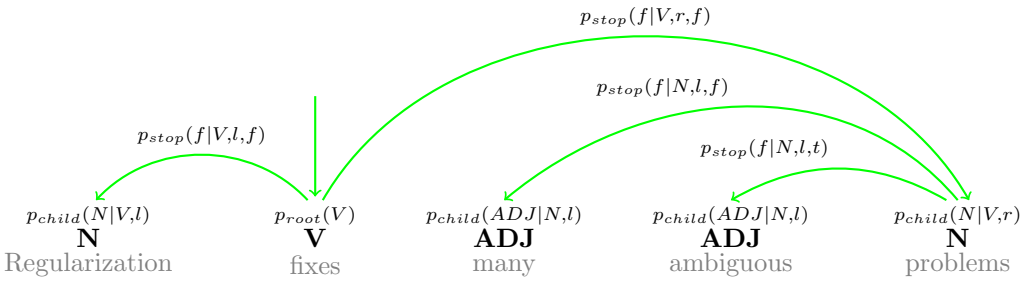


Figure 10.1: Example of a dependency tree with DMV probabilities. Right-dependents of a head are denoted by r, left-dependents by l. The letters t and f denote 'true' and 'false.' For example, in $p_{stop}(f \mid V, r, f)$ the f to the left of the conditioning bar indicates that the model has decided *not* to stop, and the other f indicates V does *not* yet have any right dependents. Note that the $p_{stop}(t \mid \ldots)$ are omitted in this diagram.

10.1.1 Dependency Model With Valence (DMV)

The DMV model specifies the following generative process. For a sentence consisting of POS tags $\mathbf{x}$, the root head POS $r(\mathbf{x})$ is generated first with probability $p_{root}(r(\mathbf{x}))$. For example, in Figure 10.1 this corresponds to generating the V with probability $p_{root}(V)$.

After generating the root, the model next generates dependents of the root. First, it generates right dependents. It decides whether to produce a right dependent conditioned on the identity of the root and the fact that it currently has no other right dependents. In our example, this decision is represented by the probability $p_{stop}(f \mid V, r, f)$. [1] If it decides to generate a right dependent, it generates a particular dependent POS by conditioning on the fact that the head POS is $r(\mathbf{x})$ and that the directionality is to the right. In our example, this corresponds to the probability $p_{child}(N \mid V, r)$. The model then returns to the choice of whether or not to stop generating right dependents, this time conditioned on the fact that it already has at least one right dependent. In our example, this corresponds to the probability $p_{stop}(t \mid V, r, t)$, which indicates that the model is done generating right dependents of V.

After stopping the generation of right dependents, the model generates left-dependents using the mirror reversal of the previous process. Once the root has generated all of its dependents, the dependents generate their own dependents recursively in the same manner.

[1] Here f represents "false" and r represents that we are generating children to the "right." In this case we decide not to stop (the first f) and conditions on not having already generated dependents on the right (the second f).

Note that in Figure 10.1 the leftmost dependent of the final N is generated before the other left dependent. The convention we are using here is that the model generates dependents starting with the rightmost one, moving inward (leftward) until all right dependents are added, then it generates the leftmost left dependent and moves inward (rightward) from there. This convention has no effect on the final probability of a parse tree under the basic DMV. However, as we will see in the following subsection, it does affect dependency tree probabilities in the extended model.

10.1.2 Model Extensions

We implemented three model extensions, borrowed from McClosky [2008] and Headden III et al. [2009]. The first extension relates to the stop probabilities, and the second two relate to dependent probabilities.

Extending Stop Probabilities

This extension conditions whether to stop generating dependents in a given direction on a larger set of previous decisions. Specifically, the probability of stopping in a particular direction depends not only on whether there are any dependents in that direction already, but also on how many. In the example of Figure 10.1, this corresponds to changing $p_{stop}(f \mid V, r, f)$ to $p_{stop}(f \mid V, r, 0)$ and similarly for all the other stop probabilities. The 0 in this case indicates that V has no other right dependents when it decides whether to continue generating right dependents.

In later sections of this chapter, when we talk about a model with maximum stop valency S, this means we distinguish the cases of $0, 1, \ldots, S-1$, and $\geq S$ dependents in a given direction. The basic DMV has maximum stop valency 1 because it distinguishes between having zero dependents and at least one dependent in a given direction. A model with maximum stop valency of 2 would distinguish between having 0, 1, or at least 2 dependents in a particular direction. In this case, when a head generates more dependents in a particular direction after its second dependent, the stopping distribution it draws from will always be the same—for head p and direction d this will be $p_{stop}(\cdot \mid p, d, 2)$.

Extending Dependent Probabilities

The second model extension we implement is analogous to the first, but applies to dependent tag probabilities instead of stop probabilities. That is, we expand the set of variables the model conditions on when selecting a particular dependent tag. Again, what we condition on is how many other dependents were already generated in the same direction. For the example in Figure 10.1, this means $p_{child}(N \mid V, r)$ becomes $p_{child}(N \mid V, r, 0)$ and similarly for all other p_{child}. In later sections of this paper, when we talk about a model with maximum child valency C, this means we distinguish between having $0, 1, \ldots, C-1$, and $\geq C$ dependents in a particular direction. The basic DMV has maximum child valency 0 because it does not make these distinctions.

This extension to the child probabilities dramatically increases model complexity. Specifically, the number of parameters grows as $O(CT^2)$. Thus, the third and final model extension we implement is to add a backoff for the child probabilities that does not condition on the identity of the parent POS (see Equation 10.2).

With this model extension, the order in which dependents are generated becomes relevant to the probability of an overall parse tree. We choose to stick with the conventional generation order described in Section 10.1.1. In cases where the identity of the rightmost and leftmost dependents have a greater influence on the true stop probability than the inner dependents, this ordering will work to the model's advantage. We do not investigate in this work which languages this holds true for, though changing this ordering might be one additional way to increase parsing accuracy for some languages.

Complete Model

Formally, under the extended DMV the probability of a sentence with POS tags $\mathbf{x}$ and dependency tree $\mathbf{y}$ is given by:

$$\begin{aligned} p_\theta(\mathbf{x}, \mathbf{y}) = & \; p_{root}(r(\mathbf{x})) \times \\ & \prod_{y \in \mathbf{y}} p_{stop}(false \mid y_p, y_d, y_{v_s}) p_{child}(y_c \mid y_p, y_d, y_{v_c}) \times \\ & \prod_{x \in \mathbf{x}} p_{stop}(true \mid x, left, x_{v_l}) \, p_{stop}(true \mid x, right, x_{v_r}) \end{aligned} \tag{10.1}$$

where $r(\mathbf{x})$ is the root tag of the dependency tree, y is the dependency of y_c on head y_p in direction y_d, and y_{v_c}, y_{v_s}, x_{v_r}, and x_{v_l} indicate valence. To formally define these last four variables, first let V_c denote the model's maximum child valency and let V_s denote maximum stop valency. Further, let a_{cpd} to be the number of y_p's dependents that are further in direction y_d than y_c, and a_{xl} (a_{xr}) be the total number of dependents of parent x to the left (right). Then we can formally express the valency variables as:

$$y_{v_c} = \min(V_c, a_{cpd}), \qquad y_{v_s} = \min(V_s, a_{cpd})$$
$$x_{v_l} = \min(V_s, a_{xl}), \qquad x_{v_r} = \min(V_s, a_{xr}).$$

In the third model extension, the backoff for the child probability to a probability not dependent on parent POS, $p_{child}(y_c \mid y_d, y_{v_c})$, can formally be expressed by:

$$\lambda p_{child}(y_c \mid y_p, y_d, y_{v_c}) + (1 - \lambda) p_{child}(y_c \mid y_d, y_{v_c}) \tag{10.2}$$

for $\lambda \in [0, 1]$. In Headden III et al. [2009] λ is a learned model parameter. In our experiments, we do not try to tune λ, but rather fix it at $1/3$. This is a crude approximation to the value used by Headden III et al. [2009] (see Section 10.2.2 for more details).

10.1.3 Model Initialization

DMV model parameter initialization plays a crucial role in the learned model's accuracy because of local maxima in the likelihood function. Klein and Manning [2004] use an "harmonic initializer", which we will refer on this paper as K&M. This initialization uses the posteriors for a fake E-step as initial parameter: posterior root probabilities are uniform $p_{root}(r(\mathbf{x})) = \frac{1}{|\mathbf{x}|}$ and head-dependent probabilities are inversely proportional to the string distance between head and dependent, $p_{child}(y_c \mid y_p, y_d, y_{v_c}) \propto \frac{1}{|y_p - y_c|}$, normalized to form a proper probability distribution. This initialization biases the parameters to prefer local attachments.

Smith [2006] compares K&M with random initialization and with uniform initialization. These two alternatives were worse than K&M unless some labeled data could be used to help pick a set of random parameters. Headden III et al. [2009] suggested using random pools to initialize the extended model described in Subsection 10.1.2 to avoid the very

strong tie between K&M and the DMV. A random pool consists of a set of B randomly initialized models trained for a small number of iterations. From these B models, the one that assigns highest likelihood to held-out development data is picked and trained until convergence. M such pools are used to create M final models, whose mean accuracy and standard deviation are reported. We will refer to this initialization method as RandomP; it performs significantly better than K&M.

Recently, Spitkovsky et al. [2010] presented several initialization methods that aim to gradually increase the complexity of the model, as measured by the size of the search space, which in the DMV model is exponential in the size of the sentence length. The Baby Steps (BS) method starts by training the model in sentences of length 1 where there is no ambiguity but nevertheless some information about heads can be gleaned. The parameters of this model are used to initialize a training run over sentences of length 2, and so on, up to a maximum length. The second method, Less is More (LsM), uses information from the BS method to pick a sentence length that includes enough sentences to train a model with good predictive power, but leaves out longer sentences that do not add much information. The hybrid method Leapfrog (LP) combines the models from the two previous approaches. All of these methods improve over the K&M initialization. In this chapter, we use K&M initialization for all experiments for simplicity. However , to achieve a fair comparison with related work, we report the accuracy differences from using different initialization methods in Subsection 10.4.3.

10.2 Previous Learning Approaches

The main comparisons for our sparse learning methods will be the expectation maximization (EM) method and Bayesian learning with a sparsity-inducing prior. We will also compare our accuracy to that achieved by several methods that use other priors. This latter comparison will be less direct though, as these priors tend to encode linguistic information at a finer-grained level. Before we make an empirical comparison in Section 10.4, in this section we review the theory behind the EM method and Bayesian learning methods.

10.2.1 Expectation Maximization

Figure 10.2 illustrates the large mismatch between an EM-trained DMV model and the empirical statistics of dependency types. We will eventually show that a parameter prior is insufficient to correct the mismatch, while posterior regularization is able to model dependency type statistics much more accurately.

10.2.2 Bayesian Learning

Recent advances in Bayesian inference methods have been applied to DMV grammar induction with varying levels of success. These approaches have focused on injecting additional linguistic intuition into the DMV by using a Dirichlet prior to sparsify parameters [Cohen et al., 2008, Headden III et al., 2009], or using logistic normal priors to tie parameters [Cohen et al., 2008, Cohen and Smith, 2009]. In the following subsections, we'll discuss how these methods work and the types of improvements they are able to achieve. We'll present a more empirical comparison with these methods later, in Section 10.4.

Sparsity-Inducing Priors

One prior that has been extensively explored for DMV learning is the Dirichlet. More precisely, a product of Dirichlets: $p(\theta) = \prod_{A \in V_N} D(\theta_A; \alpha_A)$ where we consider the DMV as a PCFG, $G = (V_N, V_T, R, S)$ with V_N, V_T, and R a set of non-terminals, terminals, and rules, respectively, and S a start symbol. A description of the rules of the DMV as a PCFG can be found in Smith [2006]. Each Dirichlet in this prior has the form:

$$D(\theta_A; \alpha_A) = \frac{1}{Z} \prod_{\beta : A \rightarrow \beta \in R} \theta_A(\beta)^{\alpha_{A \rightarrow \beta} - 1} \tag{10.3}$$

where Z is a normalization term and αs are hyperparameters.

The true posterior over the parameters, $p(\theta|\mathbf{X}) \propto \sum_{\mathbf{Y}} p(\mathbf{Y}, \mathbf{X}|\theta)p(\theta)$, is generally multi-modal and intractable to compute. The typical variational approximation is to define an approximate factored posterior over both parameters and latent variables, $q(\mathbf{Y}, \theta) = q(\mathbf{Y})q(\theta)$, and use mean field updates to minimize $\mathbf{KL}(q(\mathbf{Y})q(\theta)||p(\mathbf{Y}, \theta|\mathbf{X}))$. As shown by Kurihara and Sato [2004], with this type of prior, this can be accomplished efficiently.

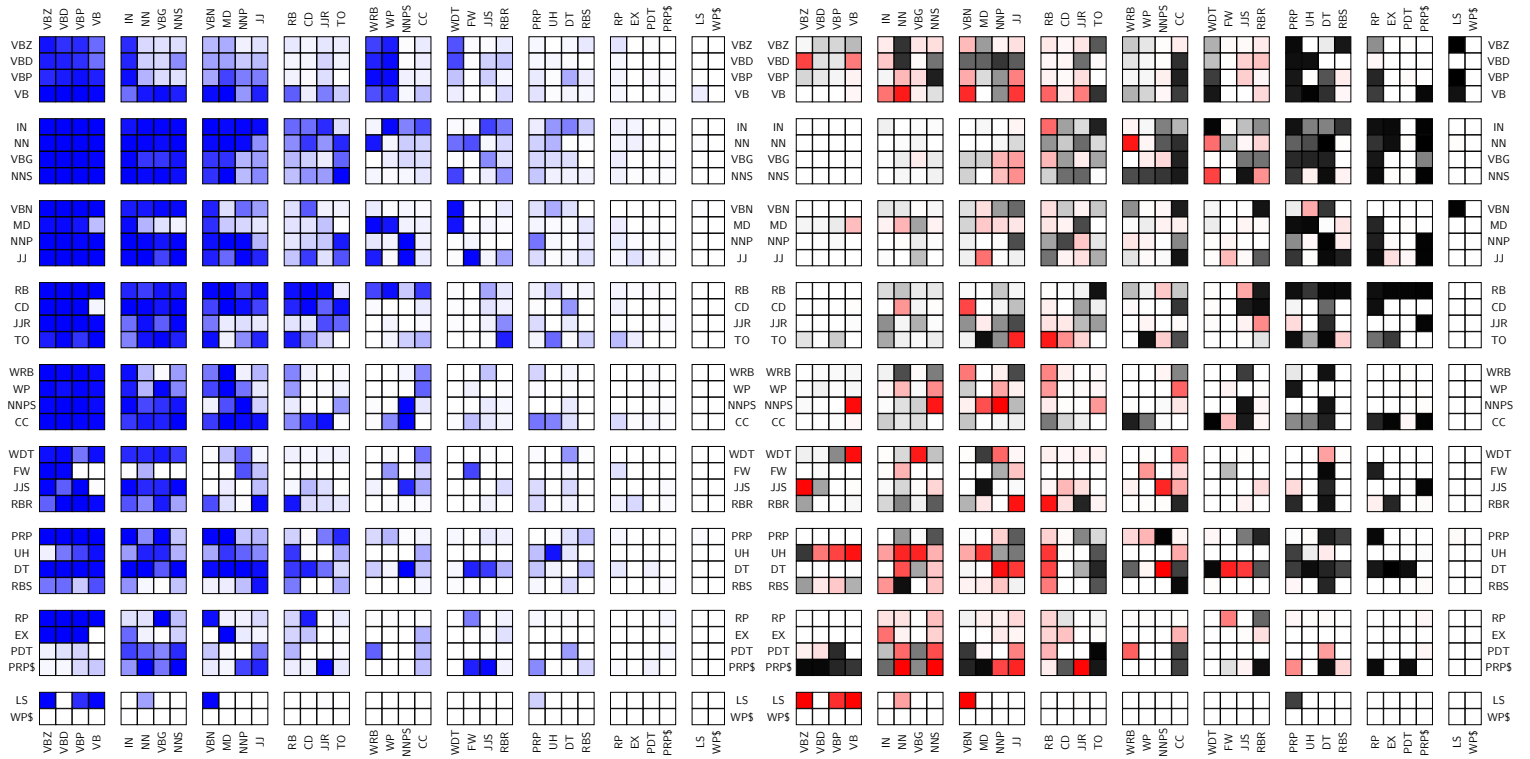


Figure 10.2: Comparison of parameters for a max likelihood DMV and an EM-trained DMV for English. Each square corresponds to a parent-child pair. Parent tags are listed across, child tags down. Parent tags are sorted left-to-right in descending order by the number of unique child tags they take. **Left**: Generated using max likelihood parameter settings (supervised). The saturation of a square with parent p and child c is determined by the maximum value of the posterior probability $p(p \mid c)$ observed in the entire English training corpus [Marcus et al., 1993]. More saturated blue indicates higher probability. **Right**: Generated using EM parameter settings. Black indicates EM posteriors are too high, red too low. More saturation indicates more deviation. White indicates no deviation. There are significantly more black squares than red, especially towards the right, indicating that EM does not learn a sparse enough model.

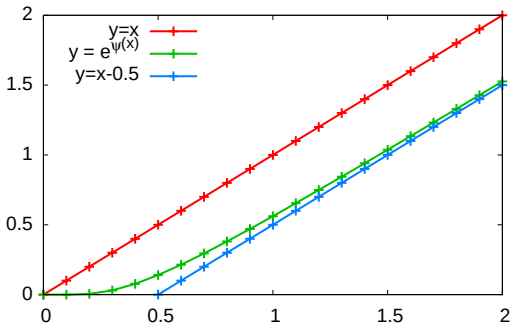


Figure 10.3: The digamma function.

Assuming the hyperparameters of the prior are fixed, the coordinate descent algorithm for updating $q(\mathbf{Y}), q(\theta)$ is similar to EM. In the E-like-step, inference for $\mathbf{Y}$ is performed using the approximate mean parameters $\bar{\theta} = \mathbf{E}_q[\theta]$. The M-like-step, is a slight modification to the standard EM M-step, both shown below:

$$\textbf{EM M-step} : \theta_A^{t+1}(\beta) \propto \mathbf{E}_{q^{t+1}}[\#_{A\to\beta}(\mathbf{Y})] \tag{10.4}$$

$$\textbf{Dirichlet M-step} : \theta_A^{t+1}(\beta) \propto \exp(\psi(\mathbf{E}_{q^{t+1}}[\#_{A\to\beta}(\mathbf{Y})] + \alpha_{A\to\beta})) \tag{10.5}$$

where ψ is the digamma function. As Figure 10.3 illustrates, $\exp(\psi(x))$ is upper bounded by $y = x$. That is, it slightly discounts the value of x, though by no more than 0.5, as $y = x - 0.5$ lower bounds it. Thus, $\exp(\psi(x + \alpha))$ is similar to adding $\alpha - 0.5$ to x. For any $\alpha < 0.5$, this encourages parameter sparsity in the **Dirichlet** M-step, since small θ will get squashed further by the digamma.

This Dirichlet prior method has been applied in several previous works. Cohen et al. [2008] use this method for dependency parsing with the DMV and achieve improvements over basic EM. They set all hyperparameters to 0.25, resulting in a sparsifying prior (this is the method referred to as VB-Dirichlet in their work). Headden III et al. [2009] also use this method to train both the DMV and the E-DMV. However, they set all hyperparameters to 1, so their prior is not aimed at sparsifying. It nevertheless produces different results than standard EM because it sets parameters according to the mean of the posterior $q(\theta)$ instead of the mode.

In this chapter we will refer to our own implementation of the VB-Dirichlet method

of Cohen et al. [2008] as the"discounting Dirichlet" (DD) method. We will show experiments applying it to both the DMV and the E-DMV. In particular we will show that while it achieves parameter sparsity, this is not the optimal sparsity to aim for in dependency parsing. Intuitively, sparsity of $p_{child}(c \mid p, d)$ means requiring that each parent tag has few unique child tags. But note that, as the supervised grid in Figure 10.2 illustrates, some parents should be allowed many different types of children. For example, VBZ, VBD, VBP, VB, IN, NN, etc. all should be able to have non-zero $p_{child}(c \mid p, d)$ for many c. We will show that posterior regularization is one way to achieve a better type of sparsity.

Parameter-Tying Priors

In addition to Dirichlet, other types of priors have been applied, namely logistic normal priors (LN)[Cohen et al., 2008] and shared logistic normal priors (SLN) [Cohen and Smith, 2009]. While the DD aims to induce parameter sparsity, LN and SLN aim to tie parameters together. Essentially, this has a similar goal to sparsity-inducing methods in that it posits a more concise explanation for the grammar of a language. That is, it suggests that POS tags share certain properties and so the grammar is not really as ambiguous as the full range of possible parameter settings would suggest.

The LN prior is the logistic transformation of a normal distribution, and can be approximated [Cohen et al., 2008] with $p(\theta) = \prod_{A \in V_N} \mathcal{N}(\mu_A, \Sigma_A)$ where μ_A is a mean vector and Σ_A is a covariance matrix for a normal distribution over the PCFG rules with lefthand side A. The Σ_A allow rules with identical lefthand sides to co-vary, effectively tying these parameters. For example, LN can tie the parameters $p_{child}(c_1 \mid p, d)$ and $p_{child}(c_2 \mid p, d)$. The SLN prior extends the capabilities of the LN prior by allowing any arbitrary parameters to be tied. In this case, parameters such as $p_{child}(c \mid p_1, d)$ and $p_{child}(c \mid p_2, d)$ can be tied even though they correspond to PCGF rules with different lefthand sides. We compare in the experimental section against some results from using LN and SLN and show that our posterior regularization method produces higher accuracy results.

As a side note, Headden III et al. [2009] also implements a sort of parameter tying for the E-DMV through a backoff distribution on child probabilities. The form of the backoff was introduced in Equation 10.2. The way Headden III et al. [2009] choose the weighting

$(1-\lambda)$ for the backoff is through a Dirichlet prior. To capture the intuition that events seen fewer times should be more strongly smoothed, this prior has hyperparameter value K for the standard child probability and value $2K$ for the backoff probability, where K is the number of PCFG rules with a particular nonterminal on the left-hand side. This ensures that the backoff probability is only ignored when enough examples of the full child probability have been seen. The prior favors the backoff 2 to 1, which is why in our approximation of this scheme we use weight $\lambda = 1/3$.

10.2.3 Other Learning Approaches

Several additional training alternatives have been proposed besides Bayesian methods. We will briefly describe three such methods here because we will compare with them in Section 10.4: contrastive estimation (CE), skewed deterministic annealing (SDA), and structural annealing (SA).

The first approach, contrastive estimation (CE), has been applied to several applications for training log linear models on unlabeled data [Smith and Eisner, 2005a,b]. The basic idea is to maximize the following:

$$\log \prod_i \frac{\sum_{\mathbf{y} \in \mathbf{Y}} \exp(\theta \cdot f(\mathbf{x}^{(i)}, \mathbf{y}))}{\sum_{(\mathbf{x},\mathbf{y}) \in N(\mathbf{x}^{(i)}) \times Y} \exp(\theta \cdot f(\mathbf{x}, \mathbf{y}))} \tag{10.6}$$

where f is some vector of feature functions, and $N(\mathbf{x}^{(i)})$ is a set of $\mathbf{x}$ that are in the "neighborhood" of $\mathbf{x}^{(i)}$. The intuition behind this method is that if a person chose to produce $\mathbf{x}^{(i)}$ out of all the possible $\mathbf{x}$ in $N(\mathbf{x}^{(i)})$, then we want to learn a model that assigns higher value to $\mathbf{x}^{(i)}$ (the numerator in Equation 10.6) than to these other $\mathbf{x}$. Restricting to a neighborhood is necessary for tractability, and the choice of neighborhood can encode linguistic knowledge. For example, for dependency parsing Smith and Eisner [2005b] formed neighborhoods by deleting any one word from $\mathbf{x}^{(i)}$, or transposing any two words.

Two other non-Bayesian approaches of note are skewed deterministic annealing (SDA) and structural annealing (SA) [Smith and Eisner, 2006]. SDA biases towards shorter dependency links as in the K&M initializer, and flattens the likelihood function to alleviate the difficulty of escaping local maxima. Alternatively, SA biases strongly toward short dependency links in early iterations, then relaxes this constraint over time.

We present an empirical comparison to the three methods from this section in Section 10.4 and show we can often achieve superior performance with posterior regularization.

10.3 Learning with Sparse Posteriors

At a high level, we would like to penalize models $p_\theta(\mathbf{Y}|\mathbf{X})$ that predict a large number of distinct dependency types. For hard assignments, this quantity is easy and intuitive to measure. Define an edge type as the pair composed of parent POS and child POS. Given a corpus with parse trees, we are interested in the number of distinct types of edges used in the labeling. Section 10.3.1 describes how to extend this definition to distributions over parse trees, by viewing it as a mixed-norm. Having a small number of distinct dependency types is a kind of sparsity structure we would like to impose on the model.

10.3.1 ℓ_1/ℓ_∞ Regularization

We now define precisely how to count dependency types, which will allow us to specify different kinds of dependeny sparsity. For each child tag c, let i range over some arbitrary enumeration of all occurrences of c in the corpus, and let p be another tag. The indicator $\phi_{cpi}(\mathbf{X}, \mathbf{Y})$ has value 1 if p is the tag of the parent of the ith occurrence of c, and value 0 otherwise. The number of unique dependency types is then given by:

$$\sum_{cp} \max_i \phi_{cpi}(\mathbf{X}, \mathbf{Y}), \tag{10.7}$$

where we sum over child-parent types cp, computing the maximum (logical or) over possible occurrences of $c \leftarrow p$ dependencies. Note that there is an asymmetry in this way of counting types: occurrences of the child type c are enumerated with i, but all occurrences of the parent type p are or-ed in ϕ_{cpi}, that is, ϕ_{cpi} is 1 if *any* occurrence of tag p is the parent of the ith occurrence of tag c, we will refer PR training with this constraint as PR-AS.

Instead of counting pairs of a child token and a parent type, we could instead have counted pairs of a child token and a parent token by letting p range over all *tokens* rather than *types*. In that case, each potential dependency would correspond to a different indicator

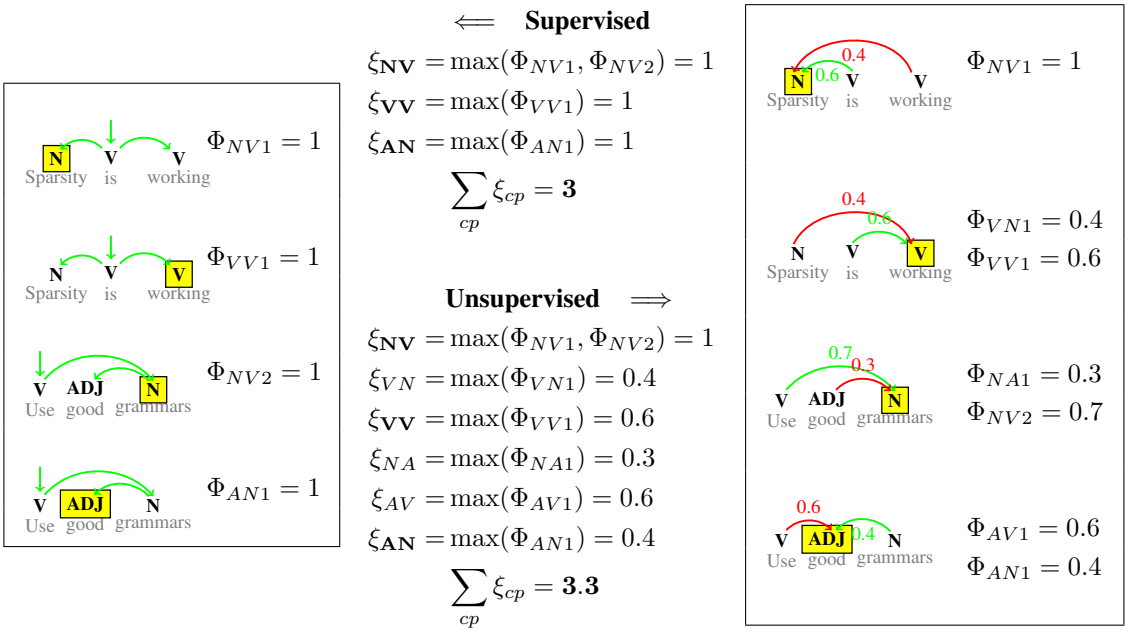


Figure 10.4: The ℓ_1/ℓ_∞ regularization term for a toy example. Let $\Phi_{cpi} = \mathbf{E}_q[\phi_{cpi}]$. For simplicity we ignore the root $\rightarrow c$ edges here, though in our experiments we incorporate their probabilities also. **Left**: Two gold parse trees with two (non-root) children each. Edges in the trees have probability 1, and all other edges probability 0, resulting in an ℓ_1/ℓ_∞ of 3. **Right**: In the unsupervised setting, instead of gold trees we have a posterior distribution over parents for each child. Given the distribution shown, the ℓ_1/ℓ_∞ is 3.3. Since real grammars tend to have few edge types, it makes sense that the ℓ_1/ℓ_∞ of a set of supervised trees will be small. Thus, using regularization to force ℓ_1/ℓ_∞ to be small for posterior distributions should push these distributions closer to the gold.

ϕ_{cpij}, and the penalty would be symmetric with respect to parents and children, we will refer PR training with this constraint as PR-S. Because of the way the model is parameterized, an asymmetry in the other direction (child-type, parent-token) does not make sense: the sum over all child types is always 1, so the penalty term would be a constant.

Both approaches perform very well, however one approach is not clearly better than the other when compared across the twelve languages. So, we report results for both versions on the results section.

Equation 10.7 can be viewed as a mixed-norm penalty on the features ϕ_{cpi}. More precisely, we will penalize the following quantity: the sum (ℓ_1 norm) over c of the maximum (ℓ_∞ norm) over occurrences of c of the posterior probability of selecting a parent with tag p for that child. Figure 10.4 shows a toy example of how to compute the ℓ_1/ℓ_∞ regularization term. The KL projection we will need to compute in order to use constrained version of the EM algorithm described in Section 2.5 is given by:

$$\arg\min_{q} \mathbf{KL}(q(\mathbf{Y})||p_\theta(\mathbf{Y}|\mathbf{X})) + \sigma \sum_{cp} \max_i \mathbf{E}_q[\phi_{cpi}(\mathbf{X}, \mathbf{Y})]. \tag{10.8}$$

Which can equivalently be written as:

$$\begin{aligned} \min_{q,\xi} \quad & \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) + \sigma \sum_{cp} \xi_{cp} \\ \text{s.t.} \quad & \xi_{cp} \leq \mathbf{E}_q[\phi_{cpi}(\mathbf{X}, \mathbf{Y})] \ \ \forall c, p, i \end{aligned} \tag{10.9}$$

where σ is the strength of the regularization, and ξ_{cp} corresponds to the maximum expectation of ϕ_{cpi} over all c and p. The dual of the projection problem is a fairly simple convex problem:

$$\begin{aligned} \min_{\lambda \geq 0} \quad & \log \left(\sum_{\mathbf{Y}} p_\theta(\mathbf{Y}|\mathbf{X}) \exp(-\boldsymbol{\lambda} \cdot \phi(\mathbf{X}, \mathbf{Y})) \right) \\ & \text{s.t.} \quad \sum_i \lambda_{cpi} \leq \sigma \end{aligned} \tag{10.10}$$

where ϕ is the vector of feature values ϕ_{cpi} for assignment $\mathbf{Y}$ of parse trees to the entire corpus $\mathbf{X}$, and $\boldsymbol{\lambda}$ is the vector of dual parameters λ_{cpi}. Note that projection onto the simplex constraints can be done very efficiently as described in Bertsekas et al. [1995].

When σ is zero, the projection is an identity mapping and the algorithm reduces to EM. As $\sigma \to \infty$, the constraints force the posterior probability of parent tag given child tag to

be uniform. For intermediate values of σ, the constraints work to decrease the confidence of the highest probability parent tags for each child instance. For parent tags that are supported by many high-probability instances, this pressure is distributed among many instances and has little effect. For parent tags that are supported by few high-probability instances however, the probability of these instances is more severely reduced, which can (after several iterations of the algorithm) effectively eliminate that parent tag as a possibility for the given child tag.

10.4 Experiments

10.4.1 Corpora

We evaluate our models on 12 languages—the English Penn Treebank [Marcus et al., 1993] and 11 languages from the CoNLL X shared task: Bulgarian [Bg] [Simov et al., 2002], Czech [Cz] [Bohomovà et al., 2001], German [De] [Brants et al., 2002], Danish [Dk] [Kromann et al., 2003], Spanish [Es] [Civit and Martí, 2004], Japanese [Jp] [Kawata and Bartels, 2000], Dutch [Nl] [Van der Beek et al., 2002], Portuguese [Pt] [Afonso et al., 2002], Swedish [Se] [Nilsson and Hall, 2005], Slovene [Sl] [Džeroski et al., 2006], and Turkish [Tr] [Oflazer et al., 2003]. For English we train on sections 2-21 of the Penn Treebank and test on section 23. In all languages, we train on the unlabeled corpora using the gold POS tags. For the other languages our train and test sets are exactly those from the CoNLL X shared task. Following the example of Smith and Eisner [2006], we strip punctuation from the sentences and keep only those sentences that are of length ≤ 10. Table 10.1 shows the size of the different training corpora after this filtering.

10.4.2 Results on English

We start with a comparison between EM and the two sparsity-inducing methods, PR and the discounting Dirichlet prior (DD), on the English corpus. For all models we use the "harmonic" K&M initializer and then train for 100 iterations. At the end of training, each model is evaluated on the test set using the Viterbi (most probable) parse. Before evalu-

		Bg	Cz	De	Dk	En	Es	Jp	Nl	Pt	Se	Si	Tr
tags		11	58	51	24	34	17	74	162	19	31	26	28
sentences	($\times 10^3$)	4.7	24	13	1.8	5.4	0.42	12	6.6	2.4	3.5	0.47	3.3
word types	($\times 10^3$)	11	40	19	5.8	10	2.6	1.9	11	7.3	7.6	2.8	10
word tokens	($\times 10^3$)	27	139	77	11	37	2.4	43	43	14	23	3.0	18

Table 10.1: Corpus statistics for sentences with lengths ≤ 10, after stripping punctuation. Bg stands for Bulgarian, Cz for Czech, De for German, Dk for Danish, En for English, Es for Spanish, Jp for Japanese, Nl for Dutch, Pt for Portuguese, Se for Swedish, Sl for Slovene, and Tr for Turkish.

		DD $\alpha =$			
	EM	1	0.25	0.1	0.01
DMV	45.8	42.2	**46.4**	45.2	45.4
2-1	45.1	42.0	46.0	45.9	44.9
2-2	**54.4**	42.0	43.3	52.5	51.5
3-3	55.3	42.8	47.1	53.5	52.1
4-4	55.1	42.9	47.1	53.6	51.7

Table 10.2: Directed attachment accuracy results on the test corpus (for sentences of lengths ≤ 10, no punctuation). The second column gives EM results, and the other columns are DD results for different settings of the hyperparameter α. The second row is for the basic DMV model, and the other rows are E-DMV models represented by their valencies (V_c-V_s). Bold represents the best parameter setting both for the DMV model and the E-DMV model.

ating, we smooth the resulting models by adding e^{-10} to each learned parameter, in order to remove the chance of zero probabilities for unseen events. (We did not bother to tune this value at all as it makes very little difference for final parses.) We score models by their attachment accuracy — the fraction of words assigned the correct parent. We compare the performance of all training procedures both on the original DMV model as well as on the extended model E-DMV. In the case of E-DMV, we set the smoothing for child probabilities to 0.66, based on the hyperparameter used in Headden III et al. [2009]. We keep smoothing fixed across languages and model configurations to reduce the number of parameters that need to be chosen. Following Cohen et al. [2008] we search for the best discounting parameter α for DD training. We tried 5 different values for α: $\{0.01, 0.1, 0.25, 1\}$.

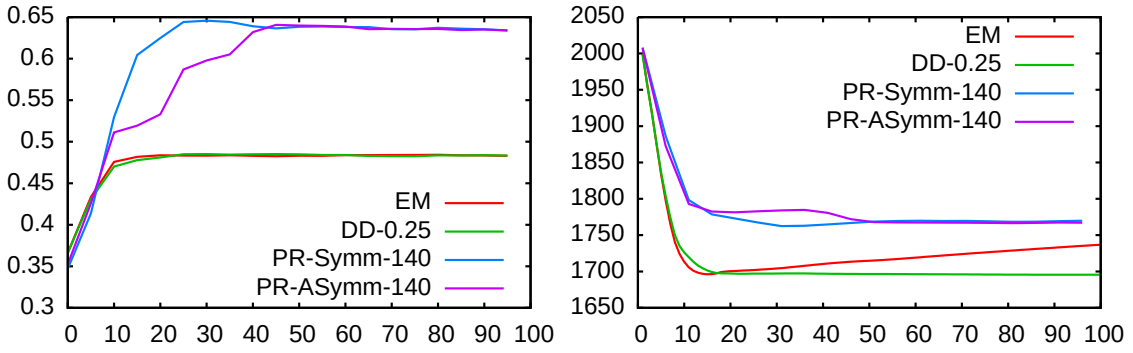
Table 10.2 shows the directed accuracy for both the DMV and the E-DMV models trained using EM and DD. We see in Table 10.2 that the extended model generally outper-

Model	EM	PR-S						PR-AS					
	DMV												
σ		80	100	120	140	160	180	80	100	120	140	160	180
	45.8	60.1	60.8	61.1	**62.1**	60.8	60.2	40.4	53.8	61.7	61.9	54.7	54.3
V_c-V_s	E-DMV												
2-1	45.1	61.1	**62.7**	62.5	62.0	61.8	60.2	40.5	54.5	61.7	62.1	54.4	62.0
2-2	54.4	**62.9**	57.3	57.4	57.4	56.7	59.2	56.2	56.3	56.8	57.0	58.5	58.7
3-3	55.3	59.4	60.4	61.1	**64.3**	63.4	62.6	60.0	60.0	61.4	63.9	64.0	59.3
4-4	55.1	61.0	62.8	64.1	63.5	64.1	59.4	59.7	59.9	60.5	**64.4**	64.1	58.1

Table 10.3: Directed attachment accuracy results on the test corpus. Bold represents the best parameter setting for the DMV model and for each of the E-DMV models. The first column contains the V_c-$V_s$used. Columns represent different σ for both constraints PR-S on the left and PR-AS on the right.

forms the DMV, for both EM and DD. However, we also see that DD does not always help: for all valences tried for the E-DMV except $(V_C, V_S) = (2, 1)$, the EM models perform better. This contrasts with the findings of Headden III et al. [2009], potentially due to the simplified smoothing that we implemented, and a difference in the stopping criterion — we ran our model for 100 iterations, while Headden III et al. [2009] ran until likelihood on a held out development set converged. Another explanation is that there are interactions of the model initialization and training. Headden III et al. [2009] use the RandomP initialization described in Subsection 10.1.3 while we use the harmonic K&M initializer. Comparing the performance of the training methods, we see that for the DMV model, DD training performs better and the best hyperparameter setting is 0.25 which is the same best parameter found by Cohen et al. [2008]. The performance of our implementation of the DD is slightly lower than the one reported in that paper, probably due to different stopping criteria during training.

A comparison between EM and PR for both DMV and E-DMV are shown in Table 10.3. We searched over six different regularization strengths (80, 100, 120, 140, 160, and 180) for both the PR-S (symmetric constraint) and PR-AS (asymmetric constraint) formulations. As with Table 10.2, the results in Table 10.3 show attachment accuracy for Viterbi decoding. In Chapter 6 we found that projecting at decoding consistently improved results for the word alignment task. For sparsity constraints on the task of dependency parsing, we found that projecting at decode time produced worse results. The regularization strength parameter σ



(a) Accuracy

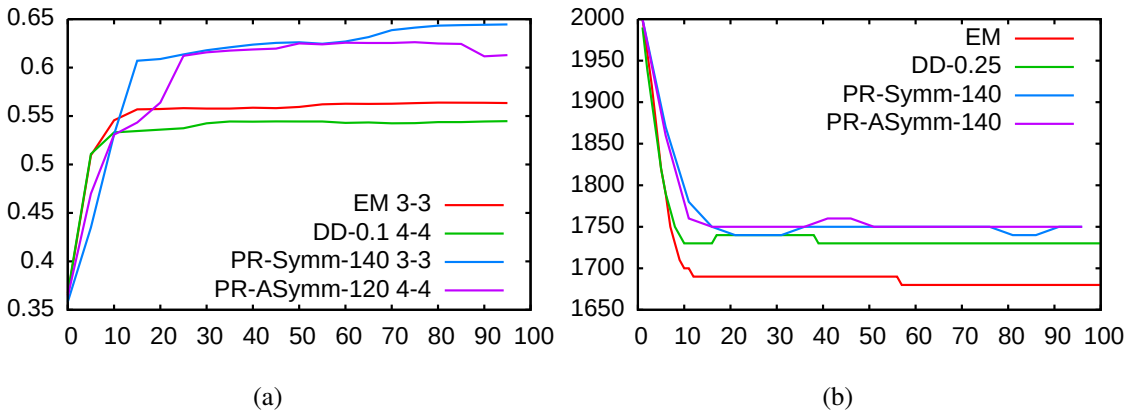
(b) Negative log-likelihood

Figure 10.5: Accuracy and negative log likelihood on held out development data as a function of the training iteration for the DMV.

is corpus-dependent and in particular depends on the size of the corpus. Because the test corpus is much smaller than the training corpus, using the same σ for training and testing might be the wrong option. Thus, in this chapter we do not project at decode time.

A first observation based on Table 10.3 is that PR-S generally performs better than the PR-AS. Furthermore, PR-S seem less sensitive to the particular regularization strength. Comparing PR-S to EM, PR-S is always better, independent of the particular σ, with improvements ranging from 4% to 16%. The PR-AS constraints are also always better than EM for each model configuration and for all but two different parameter configurations. Note that the optimal parameter σ depends on the particular model configuration (V_c-V_s).

Figure 10.5 shows how accuracy and negative log-likelihood change on a held out development corpus for the DMV. We see that both with EM and DD the models tend to converge after only 10 iterations, while for the PR training it takes 20 to 30 iterations. PR also seems to overfit and experience a degradation of performance on the test set after 40 iterations. We see in Figure 10.5 that accuracy and likelihood tend to correlate well and could potentially be used as a stopping criterion for training. In fact, most prior work uses this criterion to stop training. This appears to work for the DMV model. Figure 10.6 shows similar graphs for the E-DMV model. We see in Figure 10.6 that the likelihood-accuracy correlation does not hold as cleanly for the E-DMV model. In fact PR-AS training seem to be improving accuracy after 100 iterations, while the optimal likelihood on the held out



(a) (b)

Figure 10.6: Directed accuracy and negative log likelihood on held-out development data as a function of the training iteration for the E-DMV model with the best parameter setting.

data is achieved around iteration 20. Stopping at iteration 20 rather than 100 would have reduced accuracy by about 4%. For the rest of this paper we run all our experiences for 100 iterations and report results for the model obtained at the end of training.

Also we note that we found no correlation between the development likelihood and the best setting for the constraint strength when training with PR. This makes it harder to pick constraint strength in an unsupervised setting. We also cannot use likelihood to choose between different valencies for the E-DMV model, since the likelihoods are not comparable.

10.4.3 Comparison with Previous Work

In this section we compare the performance of different models described in the literature for unsupervised dependency parsing. Table 10.4 presents the accuracy values reported in various previous papers and the values for approaches tried in this paper. We would like to stress that the setup is not identical for all experiments. For instance, normally the stopping criteria for training is different. While we train all our models for 100 iterations, most other works use some kind of convergence criteria to stop training. Moreover, there are likely differences regarding other implementation details.

We start by comparing the effects of different initialization procedures. Although orthogonal to the learning procedure used, these differences are significant when comparing

	Init	Model	Directed			Undirected		
			≤ 10	≤ 20	all	≤ 10	≤ 20	all
Model Initialization								
1	K&M	DMV	45.8	40.2	35.9	63.4	58.0	54.2
2	RandomP	DMV	55.7(8.0)					
3	BS	Ad-Hoc @15	55.5	44.3	39.2			
4	BS	Ad-Hoc @45	55.1	44.4	39.4			
5	LsM	Ad-Hoc @15	56.2	48.2	44.1			
6	LP	Hybrid @45	**57.1**	**48.7**	**45.0**			
Smoothing effects								
7	RandomP	E-DMV(2,1) (smoothed)	**61.2(1.2)**					
8	K&M	E-DMV(2,1)	45.1	38.7	34.0	62.7	56.9	52.7
DMV								
9	K&M	DD (0.25)	46.4	40.9	36.5	64.0	58.6	54.8
10	K&M	PR-Symm 140	**62.0**	**53.8**	**49.1**	70.0	**62.6**	**58.4**
11	K&M	PR-ASymm 140	61.9	53.3	48.6	**70.2**	62.3	58.1
12	K&M	LN I	56.6	43.3	37.4			
13	K&M	LN families	59.3	45.1	39.0			
14	K&M	SLN TieV	60.2	46.2	40.0			
15	K&M	SLN TieN	60.2	46.7	40.9			
16	K&M	SLN TieV & N	61.3	47.4	41.4			
17	K&M	SLN TieA	59.9	45.8	40.9			
18	K&M	CE	48.7			64.9		
19	K&M	SDA	46.7			64.3		
20	K&M	SA	51.5			67.9		
E-DMV								
21	K&M	EM 3-3	55.3	46.4	42.6	69.0	61.9	58.3
22	K&M	DD 4-4 (0.1)	53.6	43.8	39.6	67.5	59.0	54.9
23	K&M	PR-Symm 3-3 140	64.3	**57.2**	**53.3**	**69.7**	**60.7**	56.0
24	K&M	PR-ASymm 4-4 140	64.4	55.2	50.5	69.0	60.7	**56.4**
25	K&M	EM 2-2	56.5			**69.7**		
26	RandomP	DD 2-2 (1)	53.3(7.1)					
27	RandomP	DD 2-2 (1) smoothed-skip-val	62.1(1.9)					
28	RandomP	DD 1-1 (1) smoothed-skip-head	**65.0(5.7)**					

Table 10.4: Comparison with previous published results. Results for entries 3, 4, 5, and 6 are taken from Spitkovsky et al. [2010], entries 2, 7, 26, 27, and 28 are taken from Headden III et al. [2009], entry 25 is taken from McClosky [2008], entries 12 and 13 are taken from Cohen et al. [2008], entries 14, 15, 16, and 17 are taken from Cohen and Smith [2009] and entries 18, 19, and 20 are taken from Smith [2006]. See section text for details of the comparison.

to previous work. We compare the results for the DMV with the different initializations described in Section 10.1.3. All the different approaches significantly beat the K&M initialization by about 10% in accuracy. There are some differences in the setup of the different approaches: the model initialized with RandomP described in Headden III et al. [2009] is trained using DD with a hyperparameter of 1, while all the other models are trained using EM. Additionally, the models from Spitkovsky et al. [2010] use a larger amount of data.

The next comparison we make is between the smoothing approach described in Headden III et al. [2009] and the simpler implementation done in this work. Again, although the training methods and the initialization differs we see that the smoothing performed by Headden III et al. [2009] probably increases the accuracy of that model by around 5.5% over our implementation of smoothing (see entries 1, 2, 7, and 8).

Entries 9 to 20 compare different training approaches for the basic DMV. Entry 9 corresponds to training the model with DD with the best hyperparameter setting. Entries 10 and 11 correspond to training with PR under the two types of sparsity constraints. Entries 12 and 13 use the logistic normal prior [Cohen et al., 2008] and we report the results from the paper using Viterbi decoding. Entries 14, 15, 16, and 17 correspond to the different shared logistic normal priors [Cohen and Smith, 2009]. These values are for MBR decoding since the authors don't report values for Viterbi decoding. This gives some advantage to these entries, since according to the authors MBR decoding always outperforms Viterbi decoding. Finally, entries 18, 19, and 20 represent the best value for the three learning approaches contrastive estimation (CE), skewed deterministic annealing (SDA), and structural annealing (SA) proposed by Smith [2006]. For these entries we report the best values found using supervised model selection. Out of all of these methods, the models trained using PR with the sparsity inducing constraints achieve the best results, the symmetric prior being the best. The results are similar to the best shared logistic normal prior when tested on sentences of length up to ten, but when tested on longer sentences the PR trained models perform significantly better then all other approaches.

The last block of results, entries 21 to 28, shows how a variety of learning methods compare on E-DMVs. Entries 21 to 24 compare our implementation of the three different learning approaches, EM, DD, and PR with both types of constraints. Model selection

in these cases is supervised, based on accuracy for the ≤ 10 test data. PR significantly outperforms the other two approaches. In particular the PR-S constraints perform the best with an average of 10% improvement over EM and DD on sentences of lengths ≤ 10, and an even bigger improvement for longer sentences. In entries 25 to 28 we also compare with the original extended model of McClosky [2008] and with the smoothed extended model proposed by Headden III et al. [2009]. The best model is the E-DMV with smoothing on the child probability as described by Headden III et al. [2009]. It beats the E-DMV trained with PR-S by a small amount. This small difference, 0.7%, is much smaller than the gains from using the random initialization and the better smoothing distribution. Thus, we believe that training the same model with random initialization, better child probability smoothing, and the PR constraints would in fact produce the best results. We leave this as future work.

Finally we would like to note that Table 10.4 doesn't report results for the papers that use extra information. Namely: Headden III et al. [2009] reports the best result published so far, 68.8, for the test set with sentences of lengths ≤ 10, when using lexical information. Also, Cohen and Smith [2009] reports accuracies of 62.0, 48.0, and 42.2 for sentences of lengths ≤ 10, sentences of lengths ≤ 20, and all sentences, respectively, when using multilingual information. This result for sentences of length ≤ 10 is equal to our best result, but is inferior to our results on longer sentences.

10.4.4 Multilingual Results

A grammar induction algorithm is more interesting if it works on a variety of languages. Otherwise, the algorithm might just encode a lot of language-specific information. In this section, we compare several models and learning methods on twelve different languages to test their generalization capabilities. We do not want to assume that a user would have parsed corpora in each language, so we do not include a supervised search over model parameters for all languages as part of the evaluation process. Consequently, we use the following setup: for each model, basic DMV and the four E-DMV complexities we experimented with in the previous sections, pick the best configuration found for English according to its accuracy on the ≤ 10 test set, and use it across the other eleven languages.

This might not select the ideal parameters for any particular language, but provides a more realistic test setting: a user has available a labeled corpus in one language, and would like to induce grammars for other languages of interest.

For the PR approach, since the ideal strength is related to corpus size, we try two different approaches. The first is to use exactly the same strength with other languages as used for English. The second approach is to scale the strength by the number of tokens in each corpus. In this case, the strength, σ_x, for a particular language was found by the following formula: $\sigma_x = \sigma_{en} * |tokens_x|/|tokens_{en}|$, where σ_{en} is the best strength for English, $|tokens_{en}|$ is the number of tokens of the English corpus, and $|tokens_x|$ is the number of tokens in language x. This scaling is an approximation that attempts to require a similar amount of sparsity for each language.

Table 10.5 shows the performance for all models and training procedures for the 12 different languages. Figure 10.7 illustrates the differences between the EM training and the different sparsity inducing training methods for the DMV. The zero line in Figure 10.7 corresponds to performance equal to EM. We see that the sparsifying methods tend to improve over EM most of the time. The average improvements are shown in the key of Figure 10.7. Figure 10.8 shows a similar comparison of the PR methods with respect to a DD learning baseline. We see in Figure 10.8 that PR is better than DD for most languages.

Figure 10.9 compares the different sparsity approaches. On the left we compare PR-S versus PR-AS without scaling. PR-AS beats PR-S in 9 out of 12 cases, though the average increase is only 1.5%. On the right we compare PR-S without scaling versus PR-S with scaling. The average improvement of the unscaled version is bigger for both constraints.

Figure 10.10 compares the differences of each training method against EM training using the E-DMV model with the best setting found for English. The results are similar to those for the DMV model with the biggest difference being that DD training performs worst. Both PR-S and PR-AS perform better than EM in most cases and the average improvement is even bigger than for the DMV model.

	Bg	Cz	De	Dk	En	Es	Jp	Nl	Pt	Se	Si	Tr
	DMV Model											
EM	37.8	29.6	35.7	**47.2**	45.8	40.3	52.8	37.1	35.7	39.4	42.3	46.8
DD 0.25	39.3	30.0	38.6	43.1	46.4	47.5	57.8	35.1	38.7	40.2	48.8	43.8
PR-S 140	53.7	31.5	39.6	44.0	62.1	61.1	58.8	31.0	47.0	**42.2**	39.9	51.4
PR-AS 140	**54.0**	32.0	39.6	42.4	61.9	62.4	**60.2**	**37.9**	**47.8**	38.7	**50.3**	**53.4**
PR-S s140	53.7	33.5	39.7	39.3	**62.1**	64.7	58.5	30.7	44.4	39.8	43.0	49.3
PR-AS s140	51.2	**34.1**	**40.0**	42.6	61.9	**67.9**	**60.2**	30.6	42.5	38.5	47.7	51.3
	Extended Model											
EM-(3,3)	41.7	48.9	40.1	46.4	55.3	44.3	48.5	**47.5**	35.9	**48.6**	47.5	46.2
DD-(4,4) 0.1	47.6	48.5	42.0	44.4	53.6	48.9	57.6	45.2	48.3	47.6	35.6	48.9
PR-S(3,3) 140	59.0	54.7	**47.4**	45.8	64.3	57.9	60.8	33.9	**54.3**	45.6	49.1	56.3
PR-AS(4,4) 120	59.0	53.2	45.4	46.3	**64.4**	56.1	**61.5**	38.3	49.8	41.3	**51.2**	**56.4**
PR-S(3,3) s140	**59.9**	**55.4**	45.5	42.7	64.3	68.3	57.9	34.0	46.5	44.4	48.1	**56.4**
PR-AS (4,4) s140	55.6	55.0	**47.4**	**46.9**	**64.4**	**70.6**	58.9	33.8	45.6	45.6	49.2	56.3
	Scaled Strengths											
σ120	89	445	246	37	120	9	138	138	48	76	11	58
σ140	103	519	287	43	140	10	161	161	56	89	13	68

Table 10.5: Attachment accuracy results. For each method we tested both the basic DMV and the E-DMV. The parameters used were the best parameters found for English. For the extended model the child-valency and stop-valency used are indicated in parentheses. **EM**: The EM algorithm. **DD**: Discounting Dirichlet prior. **PR-S**: Our method using the symmetric version of the constraints with strength parameter σ. **PR-S-s**: The same method but strength parameter scaled proportional to the number of tokens in the train set for each language. **PR-AS / PR-AS-s**: Our method with the asymmetric constraints, without and with scaling of the strength parameter. σ: The scaled weights for each corpus for the different values of the strength parameter used for English. Bold indicates the best method for each learning and model type.

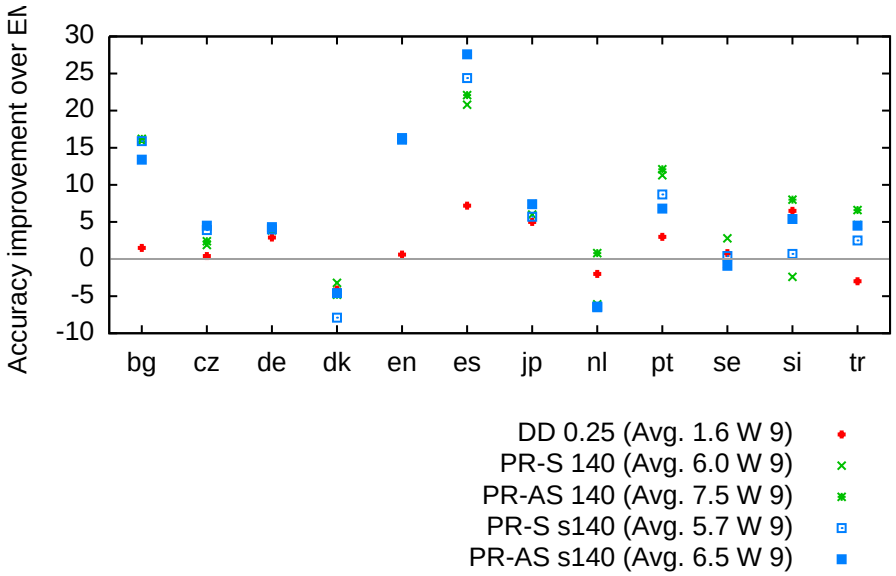


Figure 10.7: Difference in accuracy between the sparsity inducing training methods and EM training for the DMV model across the 12 languages. Avg. - Average improvement over EM. W - Number of languages better than EM.

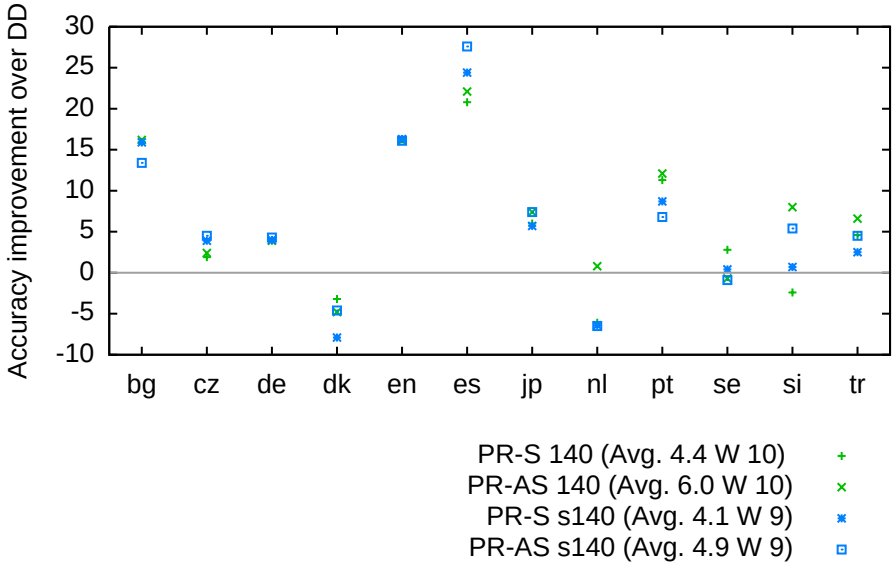
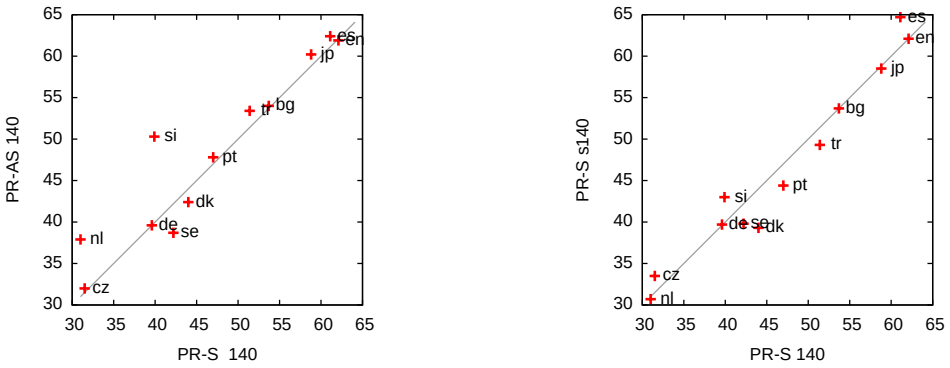


Figure 10.8: Difference in accuracy between PR training with the different constraints and DD for the DMV model across the 12 languages. Avg. - Average improvement over DD. W - Number of languages better than DD.



(a) PR-S versus PR-AS (b) No scaling versus scaling

Figure 10.9: Comparing the different sparsity constraints for the DMV model over twelve different languages. Left: PR-S vs PR-AS. Right: PR-S without scaling vs PR-S with scaling.

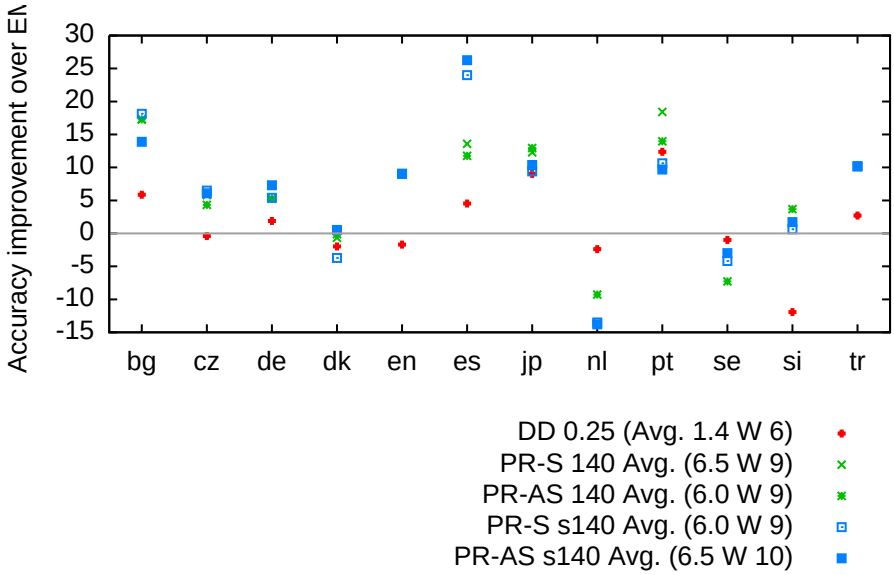


Figure 10.10: Difference in accuracy between the sparsity inducing training methods and EM training for the E-DMV model with the different training method across the 12 languages. Avg. - Average improvement over EM. W - Number of languages better than EM.

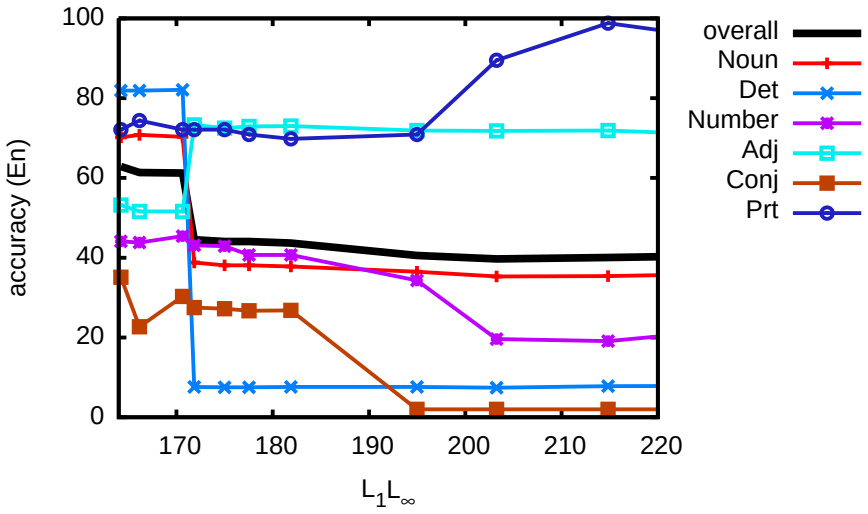


Figure 10.11: The accuracy overall and for different POS tag types in the English corpus as a function of ℓ_1/ℓ_∞ as we vary the constraint strength. EM has ℓ_1/ℓ_∞ of 431.17.

10.5 Analysis

This section attempts to describe a more qualitative picture of what the results are and where our gains and occasional losses are coming from.

10.5.1 Instability

In our experiments, we saw that the model was somewhat unstable with respect to the regularization strength. Figure 10.11 shows the accuracies on the English corpus broken down by POS tag category. The plot shows that sharp changes in overall accuracy are in fact caused by even sharper changes in the attachment accuracies of the tag categories. This should not be surprising, given that whether using EM or PR, the objective has many local maxima with deep valleys between them. The problem continues to be very underspecified, and without knowing the "true" sparsity pattern of a language, we can only achieve limited parsing accuracy.

10.5.2 Comparison of EM, PR, and DD Errors

One common EM error that PR fixes in many languages is the directionality of the noun-determiner relation. Figure 10.12 shows an example of a Spanish sentence where PR sig-

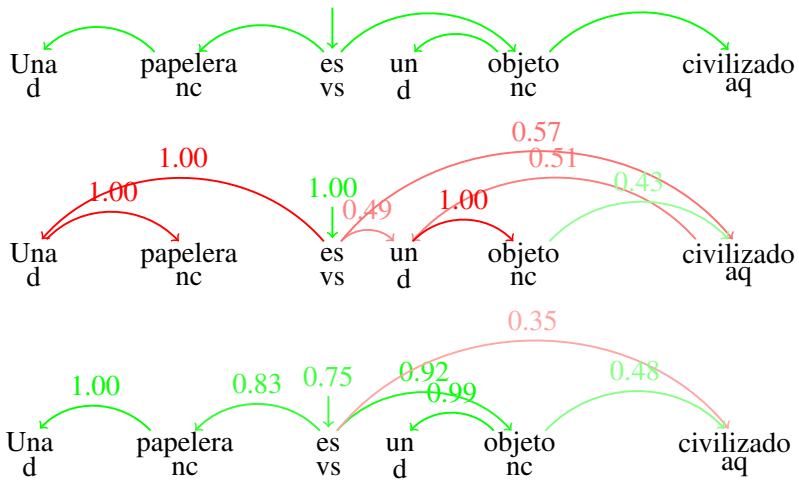


Figure 10.12: Posterior edge probabilities for an example sentence from the Spanish test corpus. **Top** is Gold, **middle** is EM, and **bottom** is PR.

nificantly outperforms standard EM because of this fixed relation. As is evidenced in this case, EM frequently assigns a determiner as the parent of a noun, instead of the reverse. PR tends not to make this error. One explanation for this improvement is that it is a result of the fact that nouns can sometimes appear without determiners. For example, consider the sentence "Lleva tiempo entenderlos" (translation: "It takes time to understand") with tags "main-verb common-noun main-verb". In this situation EM must assign the noun to a parent that is not a determiner. In contrast, when PR sees that sometimes nouns can appear without determiners but that the opposite situation does not occur, it shifts the model parameters to make nouns the parent of determiners instead of the reverse, since then it does not have to pay the cost of assigning a parent with a new tag to cover each noun that doesn't come with a determiner.

Tables 10.6 and 10.7 contrasts the most frequent types of errors EM, DD, and PR make on several test sets where PR does well. The "acc" column is accuracy and the "errs" column is the absolute number of errors of the key type. Accuracy for the key "parent POS truth/guess $\rightarrow$ child POS" is computed as a function of the true relation. So, if the key is $p_t/p_g \rightarrow c$, then accuracy is:

$$\text{acc} = \frac{\text{\# of correct } p_t \rightarrow c \text{ in Viterbi parses}}{\text{\# of } p_t \rightarrow c \text{ in gold parses}}. \tag{10.11}$$

	EM			DD			PR		
	key	**acc**	**errs**	**key**	**acc**	**errs**	**key**	**acc**	**errs**
pt	n/prp → art	0.0	39	n/prp → art	0.0	37	prp/v-fin → n	0.0	32
	v/art → n	0.0	31	v/art → n	0.0	32	n/prp → art	0.0	27
	prp/art → n	0.0	24	prp/art → n	0.0	27	v/n → prp	0.0	22
	n/v-fin → prp	0.0	18	n/v-fin → art	0.0	21	n/n → prp	0.0	20
	n/v-fin → art	0.0	17	v/v-fin → prp	72.5	11	v/prp → n	0.0	18
	v/pron-det → n	0.0	12	n/v-fin → prp	0.0	10	prp/v-fin → prop	0.0	11
	v/v-fin → prp	69.4	11	prop/prp → art	0.0	8	prp/prp → n	0.0	11
	v/prp → v	0.0	11	v/v-fin → adv	68.0	8	v/v-fin → adv	64.0	9
	prp/pron-det → n	0.0	10	prp/art → prop	0.0	7	prop/prp → art	0.0	8
	v/prp → prp	0.0	9	v/prp → v	0.0	7	v/v-fin → n	81.0	8
	prop/prp → art	0.0	8	v/prp → n	0.0	7	v/prop → prp	0.0	8
	n/v-fin → pron	0.0	8	<root>/conj-c → v	0.0	5	n/prop → prp	0.0	8
	n/prp → pron	0.0	8	v/<root> → v	0.0	5	v/v-fin → prp	58.8	7
	n/<root> → prp	0.0	8	v/art → prop	0.0	5	v/prp → v	0.0	7
	prp/art → prop	0.0	7	n/<root> → prp	0.0	5	<root>/prp → n	0.0	6
en	VB/DT → NN	0.0	129	VB/DT → NN	0.0	133	NN/NNP → NN	54.2	76
	NN/NNP → NN	60.1	65	NN/NNP → NN	54.7	78	IN/NN → NN	0.0	37
	NN/VBZ → DT	0.0	52	NN/IN → DT	0.0	56	MD/<root> → VB	0.0	25
	NN/IN → DT	0.0	47	NN/VBZ → DT	0.0	52	<root>/VB → MD	0.0	25
	IN/DT → NN	0.0	46	IN/DT → NN	0.0	46	IN/NNS → NN	0.0	24
	NN/VBD → DT	0.0	41	NN/VBD → DT	0.0	35	VB/NN → IN	0.0	21
	VB/TO → VB	0.0	19	VB/TO → VB	0.0	19	NN/NN → DT	86.5	21
	NN/VBP → DT	0.0	19	NN/VBP → DT	0.0	18	VB/DT → IN	0.0	20
	<root>/CD → NN	0.0	14	NN/NN → JJ	78.9	16	IN/VBD → NN	0.0	18
	NN/NN → JJ	81.1	14	VB/IN → JJ	0.0	12	NN/NN → JJ	79.2	16
	NN/VB → DT	0.0	14	VB/PRP$ → NN	0.0	12	IN/VBZ → NN	0.0	15
	NN/CD → CD	0.0	13	<root>/CD → NN	0.0	12	IN/VBP → NN	0.0	13
	VB/PRP$ → NN	0.0	12	NN/VB → DT	0.0	12	VB/VB → RB	18.8	13
	VB/DT → RB	0.0	11	NN/<root> → CD	0.0	11	NN/<root> → NN	0.0	11
	VB/<root> → VB	0.0	10	VB/NNS → RB	0.0	11	VB/NNS → NN	0.0	11

Table 10.6: Top 15 mistakes by parent POS truth/guess → child POS for English and the three languages where PR makes the greatest gains over EM with the E-DMV. This is the first part. Table 10.7 contains Es and Bg.

In the following subsections we provide some analysis of the results from Tables 10.6 and 10.7 for English and Bulgarian. Corrections for the other languages can be analyzed using the same type of reasoning as we have applied to analysis of English and Bulgarian. We leave interpretation of Tables 10.6 and 10.7 for Spanish and Portuguese to the reader.

10.5.3 English Corrections

There are several notable differences between EM and PR errors. Similarly to the Spanish example described above, the direction of the noun-determiner relation is corrected by

PR. This is reflected by the keys titled VB/DT → NN, NN/VBZ → DT, NN/IN → DT, IN/DT → NN, NN/VBD → DT, NN/VBP → DT, and NN/VB → DT in the English portion of Table 10.6. Recall that e.g. for VB/DT → NN the errs column of Table 10.6 counts the number of times that the model predicted that a noun (NN) should have a determiner (DT) as a parent, when in fact the true parent was a verb (VB). Since determiners can never dominate nouns, EM and DDhave accuracy 0 for this key. PR corrects this error, and propagates the change: if nouns dominate determiners, then verbs should dominate nouns rather than determiners, and prepositions should not dominate determiners in order to maintain sparsity.

A second correction PR makes is reflected in the VB/TO → VB key. One explanation for the reason PR is able to correctly identify VBs as the parents of other VBs instead of mistakenly making TO the parent of VBs is that "VB CC VB" is a frequently occurring sequence. For example, "build and hold" and "panic and bail" are two instances of the "VB CC VB" pattern from the test corpus. Presented with such scenarios, where there is no TO present to be the parent of VB, PR chooses the first VB as the parent of the second. It maintains this preference for making the first VB a parent of the second when encountered with "VB TO VB" sequences, such as "used to eliminate", because it would have to pay an additional penalty to make TO the parent of the second VB. In this manner, PR corrects the VB/TO → VB key error of EM and DD.

A third correction PR makes is reflected in the <root>/CD → NN key. This correction is similar to the noun-determiner correction: CD and NN often co-occur, but while CD almost never appears without NN, NN frequently appears without CD. Thus, if PR chose CD as parent of NN, it would have to pay an additional penalty to select another parent for NN in sentences where no CDs exist. Thus, PR is able to recognize that CD is not usually a good parent for NN. Again, EM and DD have 0 accuracy for this key.

There are a couple of errors common to EM, DD, and PR. These correspond to the NN/NN → JJ key and the NN/NNP → NN key. Because we are only controlling sparsity at the tag level, the NN/NN → JJ error is invisible to the sparsity penalty term. Similarly, since it is possible for NNP to dominate NN in some cases, the true model is not sparse with respect to removing that relation. Noun/adjective relations are notoriously difficult

to get right, especially for an unlexicalized model that also has no notion of the surface lengths of relations. We expect that combining PR with a model such as the lexicalized DMV of Headden III et al. [2009], or applying the structural annealing technique of Smith and Eisner [2006], could greatly reduce these types of errors. These changes could also help reduce some of the other main errors PR makes, such as the ones corresponding to the keys NN/NN → DT and VB/VB → RB.

Even after all these improvements, there would likely persist at least one type of English error that would be hard to fix: the domination of modals by verbs. By convention, modals dominate verbs in English dependency parses. This is a relatively arbitrary choice, as there are linguistically sound arguments to be made for either dominating the other. In fact, in some of the other languages we work with the annotation convention is the reverse of what it is in English. For now we note that the keys MD/<root> → VB and <root>/VB → MD account for a large portion of the English errors with PR.

10.5.4 Bulgarian Corrections

We might expect the results for Bulgarian to be qualitatively different than those for English for two reasons. First, the language is not in the same family as English, and in particular determiners are not separate tokens but morphologically marked. Second, we have far fewer POS tags in the Bulgarian corpus.

One large correction PR makes with respect to EM and DD corresponds to the key N/M → N.[2] This correction is similar to the English correction involving the tag CD. Another substantial correction PR makes with respect to EM and DD corresponds to the key <root>/C → V.[3] So, when we train with PR, we prefer to have verbs as sentence roots rather than as children of conjunctions. As with English determiners and nouns, the PR trained model prefers this because many sentences do not contain conjunctions. If PR chose C as the parent of V, it would have to pay a penalty to give V a different parent in single clause sentences. The same reasoning explains why PR doesn't see the V/<root> → C errors or the N/<root> → C errors that EM and DD do.

[2]N: noun, M: numeral.

[3]C: conjunction, V: verb.

	EM			DD			PR		
	key	**acc**	**errs**	**key**	**acc**	**errs**	**key**	**acc**	**errs**
es	sp/d → nc	0.0	7	sp/d → nc	0.0	7	vm/<root> → vm	0.0	5
	nc/sp → d	0.0	6	nc/sp → d	0.0	6	<root>/vm → vm	0.0	4
	vm/d → nc	0.0	5	vm/<root> → vm	0.0	6	<root>/vm → vs	0.0	3
	vs/d → nc	0.0	4	nc/vm → d	0.0	6	rg/vm → rg	0.0	2
	vm/<root> → vm	0.0	4	vm/d → nc	0.0	5	aq/aq → cc	0.0	2
	nc/vm → d	0.0	4	<root>/vm → vm	0.0	4	nc/cc → aq	0.0	2
	aq/<root> → cc	0.0	3	vs/d → nc	0.0	4	vs/<root> → vm	0.0	2
	<root>/vm → vm	0.0	3	vm/p → rn	0.0	3	aq/nc → aq	0.0	2
	vm/p → rn	0.0	3	nc/vs → d	0.0	3	vm/vm → sp	75.0	2
	nc/vs → d	0.0	3	nc/<root> → d	0.0	3	vs/vm → cs	0.0	2
	vm/nc → sp	0.0	3	vm/nc → sp	0.0	3	vm/nc → sp	0.0	2
	vm/cs → vs	0.0	2	<root>/rg → vm	0.0	2	aq/cc → aq	0.0	1
	vm/d → p	0.0	2	nc/p → d	0.0	2	nc/vs → aq	0.0	1
	nc/aq → d	0.0	2	<root>/d → nc	0.0	2	<root>/aq → nc	0.0	1
	<root>/vm → vs	0.0	2	aq/cc → aq	0.0	2	vm/vm → cc	50.0	1
bg	<root>/R → V	0.0	65	N/V → R	0.0	53	N/V → R	0.0	56
	N/<root> → R	0.0	37	V/R → N	0.0	47	V/R → N	0.0	46
	V/<root> → R	0.0	29	<root>/C → V	0.0	26	T/V → V	0.0	26
	V/R → R	0.0	24	V/R → R	0.0	25	V/R → R	0.0	25
	N/M → N	0.0	20	T/V → V	0.0	23	V/V → T	42.4	19
	V/V → T	40.6	19	N/M → N	0.0	20	N/N → N	73.4	17
	<root>/C → V	0.0	18	V/V → T	42.4	19	V/V → N	84.8	14
	V/<root> → C	0.0	17	V/<root> → C	0.0	17	V/V → C	30.0	14
	T/V → N	0.0	17	N/<root> → C	0.0	15	T/V → N	0.0	13
	N/<root> → C	0.0	16	R/N → N	0.0	14	<root>/V → T	0.0	11
	V/R → N	0.0	16	T/V → N	0.0	13	N/V → V	0.0	10
	<root>/T → V	0.0	15	V/N → N	0.0	11	T/V → P	0.0	10
	N/V → R	0.0	15	N/R → N	0.0	10	N/N → M	66.7	10
	T/<root> → V	0.0	12	V/V → N	87.3	10	V/N → N	0.0	10
	R/N → N	0.0	12	N/V → V	0.0	10	<root>/V → V	0.0	9

Table 10.7: Top 15 mistakes by parent POS truth/guess → child POS for English and the three languages where PR makes the greatest gains over EM with the E-DMV. This is the second part, Table 10.6 contains the second part containing Pt and En.

Although PR is able to make great improvements for Bulgarian parsing, it is clearly crippled by the small number of POS tags. EM, DD, and PR all make substantial errors in deciding which verb to use as the parent of a particle (see key V/V → T), and many of the main remaining errors for PR are caused by similar symmetries (see keys N/N → N, V/V → N, V/V → C, N/N → M, and <root>/V → V). As mentioned in the analysis of English, lexicalization or incorporation of a notion of surface length of relations might help alleviate these problems.

10.6 Chapter Summary

In this chapter we presented two ways of using an ℓ_1/ℓ_∞ penalty to term along with PR training to favor posterior distributions that have a small number of unique parent-child relations. The two constraints correspond to a symmetric constraint similar in spirit to the sparsity constraint applied to part-of-speech (POS) induction in Chapter 9, and an asymmetric version of the same constraint that more directly tries to minimize the number of different parent-child types instead of different parent-child occurrences. On English our approach consistently outperforms the standard EM algorithm and the approach of training in a Bayesian setting where a discounting Dirichlet prior is used.

Further, we perform an extensive comparison with previous published work and show that our learning approach achieves state-of-the-art results. We compare our approach on 11 additional languages, which as far as we know is the most extensive comparison made for dependency grammar induction. We report significant improvements over the competing learning approaches. The new approach beats EM training for 10 out of 12 languages with an average improvement of 6.5%. It also beats the Bayesian learning approach for 9 out of 12 languages with average improvement of 4% to 5%.

Unlike the ℓ_1/ℓ_∞ penalty term for POS induction, the model performance is sensitive to the strength of regularization used. The problem of choosing the correct strength in an unsupervised manner remains unsolved. We did not find a good principled solution that worked for all languages but we ruled out likelihood on held out development data as a potential candidate. However, choosing a strength on one language and applying it to others seems to work in many cases and we based our experiments on this setting.

Choosing the complexity parameters for the E-DMV model also remains an open problem. As future work we intend to investigate additional unsupervised measures for quality of dependency parses, following the recent work of Reichart and Rappoport [2009]. Even in the absence of a good unsupervised measure of model quality, a better formula for transferring the regularization strength parameter from one language to another is also needed. The regularization strength is strongly dependent on the corpus, both on the number of parent-child pairs being constrained as well as on the number of tokens for each parent and

child. Our experiments approximated this dependence by scaling the best English regularization strength by the number of tokens in other corpora, but we believe a better solution can be found.

With respect to model initialization, the K&M initialization is highly biased to the simple DMV model, and both RandomP initialization and the initialization approaches proposed by Spitkovsky et al. [2010] can significantly boost the performance of the model. We wish to initialize our models with the approaches proposed by Spitkovsky et al. [2010], since besides producing better results, those approaches are deterministic and reduce the number of parameters that need to be tuned. Following the spirit of these initialization approaches, we also propose that some success might be had by initializing the simple DMV training it, and then using its learned parameters to initialize more complex models (E-DMV models with larger valence values). This is similar in spirit to the methods for training unsupervised word alignment models.

Chapter 11

Conclusion

In this thesis we have presented posterior regularization (PR), a technique for regularizing models by encoding prior knowledge in constraints on model posteriors. On the algorithmic side, we have shown that PR can be solved efficiently in dual form, and that the regularization can be easily incorporated into a variant of the classical EM optimization method. In relating PR to similar frameworks, we have clarified its main advantages: faster optimization speed with respect to generalized expectation [Mann and McCallum, 2007, 2008], and greater distributional estimation accuracy with respect to constraint-driven learning [Chang et al., 2007]. To the best of our knowledge, we are the first to link all these learning frameworks by explicitly stating a sense in which they all approximate the Bayesian perspective that motivates Liang et al. [2009].

In addition to discussing PR's theoretical potential, we have demonstrated that it lives up to this potential in a wide variety of realistic applications. The applications we focus on in this dissertation are word alignment, multi-view learning, dependency parsing, and part of speech tagging.

11.1 Future Directions

To conclude the thesis, we describe a few possible directions for future work. Section 11.1.1 describes a few possible future applications, Section 11.1.2 suggests an empirical comparison of PR, GE and CoDL and Section 11.1.3 suggest considering possibilities

for new types of constraints. Finally Section 11.1.4 points out two areas where theoretical understanding of PR could be improved.

11.1.1 New Applications

PR can express a wide variety of prior knowledge that can be encoded by functions of model posteriors, and there remains a vast array of unexplored possible applications for this technique. Here we address a few such applications to inspire future PR work.

First of all, PR could be applied to multilingual learning, along the lines of the work of Snyder and Barzilay [2008], Snyder et al. [2009b,a]. For instance, simultaneously learning parsers for many languages, given parallel texts. This task would be similar to the dependency grammar transfer of Chapter 8, but would not require as much supervision in the sense that no language would be assumed to have a parser to begin with. The logical PR constraint in this multilingual learning setting would be that, for pairs of aligned words, the dependency relation should be present in both languages or absent in both languages a high percentage of the time.

Another important future direction for PR applications is in fields outside natural language processing. For example, consider the problem of trying to use visual data to cluster faces by character in a TV episode, as is explored by Cour et al. [2009]. In this case, we might use the prior knowledge that a face usually stays in the same horizontal position during a scene to construct a PR constraint: expected proportion of times horizontal continuity rule is broken should be less than some small value ϵ.

A final further set of applications we foresee for PR is in multi-view learning. Learning with more than two views is one direction for exploration. Chapter 7 presents many experiments with PR in the two-view setting that could easily be extended to the many-view setting, weighing each view differently as in logarithmic opinion pools. Another direction that deserves exploration with multi-view PR is problems where the proposal distribution cannot be computed in closed form, such as when combining a dependency parser with a phrase structure parser. Finally, it would be interesting to investigate under what conditions two-view PR can be expected to work. For example, under what conditions should we expect the two models to converge asymptotically faster than a monolithic model?

11.1.2 Empirical Comparison

An interesting avenue for future work includes an exploration of the trade off between computational complexity and accuracy of the different approximations presented in this and related work (Figure 4.2). For example, is there a large performance drop as we go from GE to PR and from PR to CODL, or are the variational and MAP approximations accurate in practice? Similarly it would be interesting to know whether this is application dependent. It might be the case that this depends on the accuracy of the constraints. For example, maybe constraints that are not satisfied or nearly satisfied by the true distribution are more tolerant of approximations, while having constraints that are satisfied by the truth might allow us to benefit more from GE than from PR, especially if $\mathcal{Q}$ is small.

11.1.3 New Kinds of Constraints

A last key extension to the current PR work is to explore the case where the constraint set $\mathcal{Q}$ is not easily specified using linear constraints on some constraint features ϕ. Thus far we have only developed theory and applications for linear constraints. It would be interesting to explore applications and derive efficient learning methods when the constraints are not linear, for example, applications with semi-definite or polynomial constraints. Such constraints might not permit the minimizer of the Kullback-Leibler divergence to retain the same form as the model, but perhaps under some assumptions efficient inference would still be possible.

An alternative avenue for specifying new kinds of constraints would be to abandon efficient exact inference and instead consider approximations in order to compute and use the Kullback-Leibler minimizer $q^*(\mathbf{Y})$. Bellare et al. [2009] describe some experiments with non-factorizable constraints and Monte-Carlo inference. An understanding of what kinds of constraints allow sampling in a sufficiently short amount of time is one area for future research.

11.1.4 Theoretical Analysis

For supervised learning, the probably approximately correct model of learning gives us a theoretical understanding of generalization. For example, we know that as we increase the power of a function class we wish to learn we need a larger training set in order to have high confidence of probably selecting an approximately correct function from that class. However, the situation is manageable: we only need logarithmically more examples for finite classes and for infinite classes only linearly in the Vapnik-Chervonenkis dimension of the function class.

There is no similar theory for the generalization performance of PR, GE or CoDL. It is not clear how much unlabeled data we need so that we can expect that constraints we that are satisfied on the training data will be satisfied on a new sample.

In addition to sample complexity, it would be useful to know how much prior knowledge we need for a particular problem. For example, if the constraint set contains only a few distributions of labels for a large unlabeled sample, then it seems intuitively that we should not be able to get much benefit by adding additional prior knowledge in the form of additional constraints.

Part III

Appendices

Appendix A

Probabilistic models

Throughout this thesis we are interested in estimating the parameters of probabilistic models. These are models of some quantities of interest that define a probability distribution over several outcomes. Perhaps the simplest example is a coin-flip: for some particular coin, we want to be able to predict whether it will land "heads" or "tails" when we flip it. In reality, the process of flipping a coin is very complicated: it involves a physical environment which might be changing, a precise time at which the coin is flipped, and a person performing the action. Overwhelmed by this incredibly complicated system, we typically create a very simplistic model: every time a coin is flipped, the "heads" vs. "tails" outcome is a random event, which we represent with a random variable. Let $y \in \{\text{heads}, \text{tails}\}$ be our notation for this random variable, which we with a single free parameter $0 \leq p(y = \text{heads}) \leq 1$, known as the "heads" probability. We assume that with probability $p(y = \text{heads})$, the coin will land "heads" and with probability $p(y = \text{tails}) = 1 - p(y = \text{heads})$ it will land "tails." The probability of all other outcomes — the coin landing on its side and balancing, becoming wedged vertically, being stolen by a passer-by before it reaches the ground — are assumed to be zero. Furthermore, we assume that this probability $p(y = \text{heads})$ is the same every time we flip the coin. This assumption allows us use observations of past coin flips to try to predict future coin flips. In the machine learning jargon, coin flips are assumed to be independently, identically distributed (IID). This means that the outcome of future coin flips do not depend on the outcome of previous ones, and it means that all the coin flips have the same "heads" probability.

Obviously, with a real coin in a real environment these assumptions are violated, but without making them it would be very hard to make any progress. Additionally, for the case of flipping a coin, the cost of trying to make a more complicated and powerful model outweigh the potential benefits. For phenomena where we can get more traction with relatively moderate increases in model complexity we often have much more complicated models. However, we are always forced to make simplifying assumptions, even when we know that they will be grossly violated. A goal of this thesis is to present a way to include information that we have about the real world so that the simple models we have at our disposal can more effectively predict phenomena of interest.

A.1 Latent Variables, Generative and Discriminative Models

In the coin-flip example discussed above, we have a fully observable model. When some of the random variables are hidden from observation, we might create what is called latent-variable model. For example, suppose that we cannot actually see the coin flip directly, but have the result is communicated to us through a noisy channel. Because of noise in the channel there is some probability that a result of "heads" will be communicated as "tails" and vice-versa. Let $x \in \{\text{heads}, \text{tails}\}$ be a random variable representing whether we receive "heads" or "tails" from the communication channel. We say that x is the observed variable, while y is the latent variable.

A *generative* model for this scenario would have three free parameters: the original heads probability $p(y = \text{heads})$, as well as two free parameters for the probabilities that describe the noisy channel $p(x|y)$: the probability we receive heads give the coin landed tails and the probability that we receive heads given the coin landed tails.

Potentially, we might not be interested in this full model, but might only be interested in making a decision about y given an observation x. A *discriminative* model for the system would directly model $p(y|x)$: the probability that the coin landed "heads" vs "tails" given what communication we receive from the noisy channel. This model now has only two free

parameters: $p(y = \text{heads}|x = \text{heads})$ and $p(y = \text{heads}|x = \text{tails})$. Discriminative probabilistic models are also known as conditional models. Chapter 2 shows how our framework for including prior knowledge can be used with both generative and discriminative models.

A.2 Estimation and Priors

So far we have not discussed how to use our observations, knowledge and intuition to choose the free parameters of our model. To start with, suppose that we want to base the choice entirely on a large number of observations. In this case, the most common method for choosing the model parameters is the principle of maximum likelihood [Aldrich, 1997]. The principle states that from all possible settings of the parameters of our model, we should choose the set of parameters that make our data as likely as possible. That is, we want to maximize the model's estimate of the probability of the particular set of observations that comprise our data. In the coin-flipping example from above, the maximum likelihood estimate has a closed form solution: the "heads" probability should be proportional to the number of "heads" observations. For more complicated models, we might need to estimate maximum likelihood parameters numerically by solving a possibly non-convex optimization problem.

In practice, we often have some knowledge or intuition in addition to the observations we have gathered. For example, we might know that coins "tend to be fair." The question, of course is how to encode this information. The Bayesian view on this problem is to treat the parameters of our model as also being random variables and expressing our knowledge as prior probability distributions over these parameters. After making some observations, we can infer a posterior probability distribution over the model's parameters. An important distinction between the "frequentist" and "Bayesian" views of statistics is the following: In the frequentist view, modeling a proposition as a random variable assumes that we believe it is a random process with some associated event space. By contrast, in the Bayesian view, the distribution of model parameters encodes our belief in different possible parameter values, rather than in some real-world event corresponding to model parameters. [1] Treat-

[1] The question "What is the probability that the moon is made of cheese?" makes sense from a Bayesian

ing model parameters as random variables is a very powerful technique and has become very popular in the machine learning community. However, it can also lead to parameter estimation and inference problems that are not tractable computationally, and require approximations such as Monte-Carlo methods in order to estimate the posterior distributions of model parameters. Even if we fix the model parameters, We might still need to use an approximation in order to perform inference in our model. One solution is to approximate the distribution over model parameters by its mode, choosing a single maximum a posteriori probability (MAP) setting for the parameters. Even so, the requirement of efficient computation often limits the kind of prior knowledge we can encode as Bayesian priors. In this thesis we present a different approach to encoding prior knowledge, but we show in Chapter 4 that it can be viewed as an approximation to a Bayesian approach proposed by Liang et al. [2009].

A.3 Structured Models

The coin flip example introduced earlier represents a model over an unstructured event space. There are two possible outcomes, and they are atomic. When dealing with more complicated objects, the number of outcomes quickly becomes too large for us to treat them atomically. For example, if we want to model (very small) 640×480 8-bit gray scale images, the number of possible outcomes are $8^{311\,040}$. Treating these as separate atomic entities would not be feasible, since we could not store even one parameter per possible output, and could never hope to obtain enough data to estimate our model's parameters. Similarly, the number of possible very short news articles is unmanageably large, and the number of possible segmentations of a DNA sequence into genes, even for a very simple organism is astronomical. In order to cope with this combinatorial explosion, researchers have introduce structured models that aim to decompose the distribution over possible outcomes.

As a simplistic example, consider the following generative story. Suppose that someone

point of view. If we think the moon is not made of cheese, it will be a small probability. The question does not make sense from a frequentist point of view – the moon is either made of cheese or it is not. There is no random process involved.

had access to two coins, a nickel and a dime, with different heads probabilities. They start with the nickel and repeat the following process n times:

1. throw the current coin twice and
2. write down (output) the first outcome.
3. If the second outcome is "tails" then switch coins, otherwise keep the same coin.
4. In either case, return to step 1.

If we want to model the distribution over possible outputs, then we have to model a sequence of n binary flips and there are 2^n possible outcomes $\mathbf{x} = (x_1, \ldots, x_n)$. However, maintaining $O(2^n)$ parameters is very wasteful, since if we know that two free parameters are enough to fully specify the process: the heads probability of the nickel and the heads probability of the dime. Here, the latent variable is the sequence of current coins $\mathbf{y} = (y_1, \ldots, y_n)$ where $y_i \in \{\text{nickel}, \text{dime}\}$.

The process described above is known as a hidden Markov model Rabiner [1989], and we will use them again in the running example of Chapter 2. This is a structured model because we can decompose the distribution $p(\mathbf{x}, \mathbf{y})$ as a product over some parts:

$$p(\mathbf{x}, \mathbf{y}) = \prod_i p(y_i|y_{i-1})p(x_i|y_i). \tag{A.1}$$

It is also possible to have structured conditional models, such as conditional random fields [Lafferty et al., 2001].

A.4 Maximum Entropy and Maximum Likelihood

In Chapter 2 we use a convex duality in order to efficiently deal with factorizable prior information in structured models. That duality is very similar to a well-known duality between maximum likelihood and maximum entropy. The idea behind the maximum entropy principle is the following. Suppose that we have decided on a set of statistics that are sufficient for our problem. That is to say, we have identified some set of features $\mathbf{f}(y)$, and we have collected their expected values $\tilde{\mathbf{f}} = \mathbf{E}[\mathbf{f}(y)]$ from some number of observations. In the

coin-flipping example, $\mathbf{f}(y)$ might be just a function that has value 1 when the coin lands "heads" and 0 when it lands "tails." Then $\tilde{\mathbf{f}}$ will be the fraction of "heads" in our sample. In general we might collect many statistics, millions in the case of some natural language processing applications, and $\tilde{\mathbf{f}}$ will be a vector of empirical averages. We can use as $\mathbf{f}(y)$ any functions whose values we can compute. Given the statistics, we want to know how to choose a model $p(y)$ or a conditional model $p(y|x)$. Naturally we would like the expectations of the features $\mathbf{f}(y)$ under the model to be close to the values we observed $\tilde{\mathbf{f}}$, however for most applications there are infinitely many such models. The principle of maximum entropy [Jaynes, 1957, Good, 1963] states that we should choose the model $p(y)$ that has highest entropy among all models satisfying the constraints that $\mathbf{E}_p[\mathbf{f}] \approx \tilde{\mathbf{f}}$. Writing this out as a maximization problem, we would like to solve:

$$\max_p \quad \mathbf{E}_p\left[-\log p(y)\right] \qquad \text{s.t.} \quad ||\mathbf{E}_p[\mathbf{f}(y)] - \tilde{\mathbf{f}}||_\beta \leq \epsilon \tag{A.2}$$

where $||\cdot||_\beta$ is a norm, which along with ϵ encodes how "far" our model's feature expectations are allowed to be from the empirical averages. Note that so far we have not stated what form the model p should have. If y lives in a continuous space, then it might not be clear from Equation A.2 that the maximizer of the equation — $p^*(y)$ — is even representable efficiently. However, by taking the dual of the optimization problem in Equation A.2 we can see that $p^*(y)$ can be represented as:

$$p^*(y) \propto \exp(\mathbf{f}(y) \cdot \theta^*) \tag{A.3}$$

where θ^* are the optimal dual parameters. The dual problem for $\epsilon = 0$ is to find the maximum likelihood of the observed data from the family of models parameterized as in Equation A.3. If $\epsilon > 0$ then the dual of Equation A.2 can be interpreted as a maximum a posteriori probability with some prior.

This duality is a special case of one we use in our framework, as described in Chapter 2. A proof for the more-general case is given in Appendix C.

Appendix B

Scaling the strength of PR

This appendix describes how to optimize a version of our objective with scaled posterior regularization strength. In this case, we will use a modified EM algorithm that maximizes:

$$F'(q, \theta) = \mathcal{L}(\theta) - \alpha \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y} \mid \mathbf{X})) \qquad \text{s.t. } q \in \mathcal{Q} \tag{B.1}$$

where $\alpha \in [0, 1]$. The optimization procedure closely follows the one in Section 2.5. When performing the M-step, we use a mixture of the projected posteriors q and the model posteriors $p_\theta(\mathbf{Y}|\mathbf{X})$ to update the model parameters. The updated EM algorithm is:

$$\mathbf{E}' - \mathbf{step}: \max_q F'(q, \theta) = \min_{q \in \mathcal{Q}} \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \tag{B.2}$$

$$\mathbf{M}' - \mathbf{step}: \max_\theta F'(q, \theta) = \max_\theta (1 - \alpha)\mathbf{E}_{p_{\theta'}}[\log p_\theta(\mathbf{X}, \mathbf{Y})] + \alpha \mathbf{E}_q[\log p_\theta(\mathbf{X}, \mathbf{Y})] \tag{B.3}$$

Note that the $\mathbf{E}'$-step is identical to the one in Equation 2.17.

In the case where $\alpha > 1$, this simple scheme will no longer work, since α is being used to control the relative weighting of the expectations according to q and those according to p_θ. In some cases, it is possible to compute the gradient or sub-gradient of Equation B.1 with respect to θ, and then Equation B.1 can be optimized with a gradient or sub-gradient ascent procedure.

Appendix C

Proof of Proposition 2.1

The modified E-step involves a projection step that minimizes the Kullback-Leibler divergence:

$$\arg\min_{q,\xi} \quad \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \qquad \text{s.t.} \quad \mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b} \leq \xi; \quad ||\xi||_\beta \leq \epsilon \quad \text{(C.1)}$$

Assuming the set $\mathcal{Q} = \{q(\mathbf{Y}) : \exists \xi : \mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b} \leq \xi; \quad ||\xi||_\beta \leq \epsilon\}$ is non-empty, the corresponding Lagrangian is

$$\max_{\lambda \geq 0, \alpha \geq 0, \gamma} \quad \min_{q,\xi} \quad L(q(\mathbf{Y}), \xi, \lambda, \alpha, \gamma), \quad \text{(C.2)}$$

where

$$\begin{aligned} L(q,\xi,\lambda,\alpha,\gamma) = \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \quad &+ \lambda \cdot (\mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b} - \xi) \\ &+ \alpha(||\xi||_\beta - \epsilon) \quad + \gamma \left(\sum_{\mathbf{Y}} q(\mathbf{Y}) - 1 \right) \quad \text{(C.3)} \end{aligned}$$

In order to compute the dual of this Lagrangian, we first represent

$$\alpha \, ||\xi||_\beta = \max_\eta \xi \cdot \eta \quad \text{s.t.} \quad ||\eta||_{\beta^*} \leq \alpha. \quad \text{(C.4)}$$

This results in a variational Lagrangian

$$\max_{\lambda \geq 0, \alpha \geq 0, \gamma} \quad \max_{||\eta||_{\beta^*} \leq \alpha} \quad \min_{q,\xi} \quad L(q(\mathbf{Y}), \xi, \lambda, \alpha, \gamma, \eta), \quad \text{(C.5)}$$

with $L(q(\mathbf{Y}), \xi, \lambda, \alpha, \gamma, \eta)$ defined as

$$L(q, \xi, \lambda, \alpha, \gamma, \eta) = \quad \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \quad + \lambda \cdot (\mathbf{E}_q[\phi(\mathbf{X}, \mathbf{Y})] - \mathbf{b} - \xi) \\ + \xi \cdot \eta - \alpha\epsilon \quad + \gamma \left(\sum_{\mathbf{Y}} q(\mathbf{Y}) - 1 \right) \tag{C.6}$$

$$\frac{\partial L(q(\mathbf{Y}), \xi, \lambda, \alpha, \gamma, \eta)}{\partial q(\mathbf{Y})} = \quad \log q(\mathbf{Y}) + 1 - \log p_\theta(\mathbf{Y}|\mathbf{X}) + \lambda \cdot \phi(\mathbf{X}, \mathbf{Y}) + \gamma = 0 \tag{C.7}$$

$$\implies \quad q(\mathbf{Y}) = \frac{p_\theta(\mathbf{Y}|\mathbf{X}) \exp(-\lambda \cdot \phi(\mathbf{X}, \mathbf{Y}))}{e \exp(\gamma)} \tag{C.8}$$

$$\frac{\partial L(q(\mathbf{Y}), \xi, \lambda, \alpha, \gamma, \eta)}{\partial \xi_i} = \quad \eta_i - \lambda_i = 0 \quad \implies \quad \eta = \lambda \tag{C.9}$$

Note that Equation C.9 implies that we have the constraint $||\lambda||_{\beta^*} \leq \alpha$ and also the positive and negative $\lambda \cdot \xi$ cancel each other out. Plugging $q(\mathbf{Y})$, $\eta = \lambda$ in $L(q(\mathbf{Y}), \xi, \lambda, \alpha, \gamma, \eta)$ and taking the derivative with respect to γ

$$\frac{\partial L(\lambda, \alpha, \gamma)}{\partial \gamma} = \sum_{\mathbf{Y}} \frac{p_\theta(\mathbf{Y}|\mathbf{X}) \exp(-\lambda \cdot \phi(\mathbf{X}, \mathbf{Y}))}{e \exp(\gamma)} - 1 = 0 \\ \implies \gamma = \log \left(\frac{\sum_{\mathbf{Y}} p_\theta(\mathbf{Y}|\mathbf{X}) \exp\left(-\lambda \cdot \phi(\mathbf{x}, \mathbf{z})\right)}{e} \right). \tag{C.10}$$

From there we can simplify $q(\mathbf{Y}) = \frac{p_\theta(\mathbf{Y}|\mathbf{X}) \exp(-\lambda \cdot \phi(\mathbf{X}, \mathbf{Y}))}{Z_\lambda}$, where

$$Z_\lambda = \sum_{\mathbf{Y}} p_\theta(\mathbf{Y}|\mathbf{X}) \exp(-\lambda \cdot \phi(\mathbf{X}, \mathbf{Y})) \tag{C.11}$$

ensures that $q(\mathbf{Y})$ is properly normalized. Plugging γ into $L(\lambda, \alpha, \gamma)$

$$L(\lambda, \alpha) = -\log(Z_\lambda) - \mathbf{b} \cdot \lambda - \alpha\epsilon \tag{C.12}$$

Now our objective is:

$$\max_{\lambda \geq 0, \alpha \geq 0} \quad -\log(Z_\lambda) - \mathbf{b} \cdot \lambda - \alpha\epsilon \quad \text{s.t.} \quad ||\lambda||_{\beta^*} \leq \alpha \tag{C.13}$$

We can analytically see that the optimum of this objective with respect to α is $\alpha = ||\lambda||_{\beta^*}$ and placing this in $L(\lambda, \alpha)$ we get the dual objective:

$$\textbf{Dual } \mathbf{E}' : \quad \arg\max_{\lambda \geq 0} \quad -\mathbf{b} \cdot \lambda - \log(Z_\lambda) - \epsilon \, ||\lambda||_{\beta^*} \tag{C.14}$$

as desired.

Bibliography

A. Abeillé. *Treebanks: Building and Using Parsed Corpora*. Springer, 2003.

S. Afonso, E. Bick, R. Haber, and D. Santos. Floresta Sinta(c)tica: a treebank for Portuguese. In *Proc. LREC*, 2002.

J. Aldrich. RA Fisher and the making of maximum likelihood 1912-1922. *Statistical Science*, 12(3):162–176, 1997.

Y. Altun, M. Johnson, and T. Hofmann. Investigating loss functions and optimization methods for discriminative learning of label sequences. In *Proc. EMNLP*, 2003.

R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

J. Atserias, B. Casas, E. Comelles, M. González, L. Padró, and M. Padró. Freeling 1.3: Syntactic and semantic services in an open-source nlp library. In *Proc. LREC*, 2006.

M. Balcan and A. Blum. A PAC-style model for learning from labeled and unlabeled data. In *Proc. COLT*, 2005.

C. Bannard and C. Callison-Burch. Paraphrasing with bilingual parallel corpora. In *Proc. ACL*, 2005.

K. Bellare, G. Druck, and A. McCallum. Alternating projections for learning with expectation constraints. In *Proc. UAI*, 2009.

A. Bernal, K. Crammer, A. Hatzigeorgiou, and F. Pereira. Global discriminative learning for higher-accuracy computational gene prediction. *PLoS Computational Biology*, 3(3): e54, 2007.

D. P. Bertsekas. *Nonlinear Programming: 2nd Edition*. Athena scientific, 1999.

D.P. Bertsekas, M.L. Homer, D.A. Logan, and S.D. Patek. Nonlinear programming. *Athena scientific*, 1995.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003. URL `http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf`.

J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proc. EMNLP*, 2006.

J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. ACL*, 2007.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. COLT*, 1998.

A. Bohomovà, J. Hajic, E. Hajicova, and B. Hladka. The prague dependency treebank: Three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*. Kluwer Academic Publishers, 2001.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, 2002.

U. Brefeld, C. Büscher, and T. Scheffer. Multi-view hidden markov perceptrons. In *Proc. LWA*, 2005.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, M. J. Goldsmith, J. Hajic, R. L. Mercer, and S. Mohanty. But dictionaries are data too. In *Proc. HLT*, 1993.

P. F. Brown, S. Della Pietra, V. J. Della Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2): 263–311, 1994.

C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of molecular biology*, 268(1):78–94, 1997.

C. Callison-Burch. *Paraphrasing and Translation*. PhD thesis, University of Edinburgh, 2007.

C. Callison-Burch. Syntactic Constraints on Paraphrases Extracted from Parallel Corpora. In *Proc. EMNLP*, 2008.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. Coupled Semi-Supervised Learning for Information Extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM)*, 2010.

M. Chang, L. Ratinov, and D. Roth. Guiding semi-supervision with constraint-driven learning. In *Proc. ACL*, 2007.

M.W. Chang, L. Ratinov, N. Rizzolo, and D. Roth. Learning and inference with constraints. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. AAAI, 2008.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning (Adaptive Computation and Machine Learning)*. The MIT Press, September 2006. ISBN 0262033585.

C. Chelba, D. Engle, F. Jelinek, V. Jimenez, S. Khudanpur, L. Mangu, H. Printz, E. Ristad, R. Rosenfeld, A. Stolcke, and D. Wu. Structure and performance of a dependency language model. In *Proc. Eurospeech*, 1997.

D. Chiang, A. Lopez, N. Madnani, C. Monz, P. Resnik, and M. Subotin. The hiero machine translation system: extensions, evaluation, and analysis. In *Proc. HLT-EMNLP*, 2005.

K Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*. Association for Computational Linguistics, 1988.

M. Civit and M.A. Martí. Building cast3lb: A Spanish Treebank. *Research on Language & Computation*, 2004.

S.B. Cohen and N.A. Smith. The shared logistic normal distribution for grammar induction. In *Proc. NAACL*, 2009.

S.B. Cohen, K. Gimpel, and N.A. Smith. Logistic normal priors for unsupervised probabilistic grammar induction. In *Proc. NIPS*, 2008.

M. Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proc. SIGDAT-EMNLP*, 1999.

T. Cour, B. Sapp, C. Jordan, and B. Taskar. Learning from ambiguously labled images. In *Proc. CVPR*, 2009.

H. Daumé III. Cross-task knowledge-constrained self training. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2008.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Royal Statistical Society, Ser. B*, 39(1):1–38, 1977.

S.J. DeRose. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14(1):39, 1988.

Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, João Graca, and Fernando Pereira. Frustratingly hard domain adaptation for dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 1051–1055, 2007. URL `http://www.aclweb.org/anthology/D/D07/D07-1112`.

G. Druck, G. Mann, and A. McCallum. Learning from labeled features using generalized expectation criteria. In *Proc. SIGIR*, 2008.

G. Druck, G. Mann, and A. McCallum. Semi-supervised learning of dependency parsers using generalized expectation criteria. In *Proc. ACL-IJCNLP*, 2009.

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. Towards a Slovene dependency treebank. In *Proc. LREC*, 2006.

J. Eisner. Three new probabilistic models for dependency parsing: an exploration. In *Proc. CoLing*, 1996.

J. Finkel, T. Grenager, and C. Manning. The infinite tree. In *Proc. ACL*, 2007.

H. Fox. Phrasal cohesion and statistical machine translation. In *Proc. EMNLP*, 2002.

M. Galley, M. Hopkins, K. Knight, and D. Marcu. What's in a translation rule? In *Proc. HLT-NAACL*, 2004.

K. Ganchev, J. Graça, J. Blitzer, and B. Taskar. Multi-view learning over structured and non-identical outputs. In *Proc. UAI*, 2008.

K. Ganchev, J. Gillenwater, and B. Taskar. Dependency grammar induction via bitext projection constraints. In *Proc. ACL-IJCNLP*, 2009a.

Kuzman Ganchev, Joao Graca, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. Technical Report MS-CIS-09-16, University of Pennsylvania Department of Computer and Information Science, 2009b.

Kuzman Ganchev, Joao Graca, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *submitted to Journal of Machine Learning Research*, 10, 2010.

J. Gao and M. Johnson. A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers. In *Proc. EMNLP*, 2008.

S. Goldwater and T. Griffiths. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proc. ACL*, 2007.

I.J. Good. Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables. *The Annals of Mathematical Statistics*, 34(3):911–934, 1963.

J. Graça, K. Ganchev, and B. Taskar. Expectation maximization and posterior constraints. In *Proc. NIPS*, 2007.

J. Graça, K. Ganchev, F. Pereira, and B. Taskar. Parameter vs. posterior sparisty in latent variable models. In *Proc. NIPS*, 2009a.

J. Graça, K. Ganchev, and B. Taskar. Learning tractable word alignment models with non-markovian constraints - in sumbission. In *Computational Linguistics*, 2009b.

J. Graça, K. Ganchev, and B. Taskar. Postcat - posterior constrained alignment toolkit. In *The Third Machine Translation Marathon*, 2009c.

S. Greenberg. The Switchboard transcription project. Technical Report 24, Center for Language and Speech Processing, Johns Hopkins University, 1996.

A. Haghighi and D. Klein. Prototype-driven learning for sequence models. In *Proc. NAACL*, 2006.

A. Haghighi, A. Ng, and C. Manning. Robust textual inference via graph matching. In *Proc. EMNLP*, 2005.

W.P. Headden III, M. Johnson, and D. McClosky. Improving unsupervised dependency parsing with richer contexts and smoothin g. In *Proc. NAACL*, 2009.

R. Hwa, P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11:11–311, 2005.

E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620–630, 1957.

M. Johnson. Why doesn't EM find good HMM POS-taggers. In *Proc. EMNLP-CoNLL*, 2007.

M. Johnson, T.L. Griffiths, and S. Goldwater. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In *Proc. NIPS*, 2007.

B.H. Juang and L.R. Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

T. Kailath. The divergence and bhattacharyya distance measures in signal selection. *IEEE Transactions on Communications*, 15(1):52–60, 2 1967. ISSN 0096-2244.

S. Kakade and D. Foster. Multi-view regression via canonical correlation analysis. In *Proc. COLT*, 2007.

Y. Kawata and J. Bartels. Stylebook for the Japanese Treebank in VERBMOBIL. Technical report, Eberhard-Karls-Universitat Tubingen, 2000.

R. Kindermann, J.L. Snell, and American Mathematical Society. *Markov random fields and their applications*. American Mathematical Society Providence, Rhode Island, 1980.

D. Klein and C. Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. ACL*, 2004.

P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*, 2005.

P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *Proc. NAACL*, 2003.

M.T. Kromann, L. Mikkelsen, and S.K. Lynge. Danish Dependency Treebank. In *Proc. TLT2003*, 2003.

K. Kurihara and T. Sato. An application of the variational Bayesian approach to probabilistic context-free grammars. In *IJC-NLP Workshop: Beyond Shallow Analyses*, 2004.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, 2001.

S. Lee and K. Choi. Reestimation and best-first parsing algorithm for probabilistic dependency grammar. In *Proc. WVLC-5*, 1997.

Zhifei Li and Jason Eisner. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 40–51, Singapore, 2009.

P. Liang, B. Taskar, and D. Klein. Alignment by agreement. In *Proc. HLT-NAACL*, 2006.

P. Liang, S. Petrov, M.I. Jordan, and D. Klein. The infinite PCFG using hierarchical Dirichlet processes. In *Proc. EMNLP*, 2007.

P. Liang, M. I. Jordan, and D. Klein. Learning from measurements in exponential families. In *Proc. ICML*, 2009.

G. S. Mann and A. McCallum. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proc. ICML*, 2007.

G. S. Mann and A. McCallum. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proc. ACL*, 2008.

G.S. Mann and A. McCallum. Generalized Expectation Criteria for Semi-Supervised Learning with Weakly Labeled Data. *The Journal of Machine Learning Research*, 11: 955–984, 2010.

C.D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, 2000.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.

E. Matusov, R. Zens, and H. Ney. Symmetric word alignments for statistical machine translation. In *Proc. COLING*, 2004.

E. Matusov, N. Ueffing, and H. Ney. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proc. EACL*, 2006.

D. McClosky. Modeling valence effects in unsupervised grammar induction. Technical report, CS-09-01, Brown University, 2008.

R. McDonald. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. PhD thesis, University of Pennsylvania, 2006.

R. McDonald, K. Crammer, and F. Pereira. Online large-margin training of dependency parsers. In *Proc. ACL*, 2005.

B. Merialdo. Tagging English text with a probabilistic model. *Computational linguistics*, 20(2):155–171, 1994.

R. C. Moore. Improving IBM word-alignment model 1. In *Proc. ACL*, 2004.

R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer, 1998.

J. Nilsson and J. Hall, J.and Nivre. MAMBA meets TIGER: Reconstructing a Swedish treebank from antiquity. *NODALIDA Special Session on Treebanks*, 2005.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proc. EMNLP-CoNLL*, 2007.

F. J. Och and H. Ney. Improved statistical alignment models. In *Proc. ACL*, 2000.

F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003. ISSN 0891-2017.

K. Oflazer, B. Say, D.Z. Hakkani-Tür, and G. Tür. Building a Turkish treebank. *Treebanks: Building and Using Parsed Corpora*, 2003.

Adam Pauls, John Denero, and Dan Klein. Consensus training for consensus decoding in machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in*

Natural Language Processing (EMNLP), pages 1418–1427, Singapore, 2009. Association for Computational Linguistics.

Novi Quadrianto, James Petterson, and Alex Smola. Distribution matching for transduction. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1500–1508. MIT Press, 2009.

C. Quirk, A. Menezes, and C. Cherry. Dependency treelet translation: syntactically informed phrasal smt. In *Proc. ACL*, 2005.

L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *In Proc. IEEE*, 77(2):257–286, 1989.

R. Reichart and A. Rappoport. Automatic selection of high quality parses created by a fully unsupervised parser. In *Proc.CoNLL*, 2009.

M. Rogati, S. McCarley, and Y. Yang. Unsupervised learning of arabic stemming using a parallel corpus. In *Proc. ACL*, 2003.

D. Rosenberg and P. Bartlett. The rademacher complexity of co-regularized kernel classes. In *Proc. AI Stats*, 2007.

E. F. Tjong Kim Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *Proc. CoNLL and LLL*, 2000a.

E. F. Tjong Kim Sang and F. De Meulder. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proc. HLT-NAACL*, 2003.

E.F.T.K. Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132. Lisbon, Portugal, 2000b.

L. Shen, J. Xu, and R. Weischedel. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. ACL*, 2008.

K. Simov, P. Osenova, M. Slavcheva, S. Kolkovska, E. Balabanova, D. Doikoff, K. Ivanova, A. Simov, E. Simov, and M. Kouylekov. Building a linguistically interpreted corpus of bulgarian: the bultreebank. In *Proc. LREC*, 2002.

V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularization approach to semi-supervised learning with multiple views. In *Proc. ICML*, 2005.

L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, 4(3):519–524, 1987.

A. Smith, T. Cohn, and M. Osborne. Logarithmic opinion pools for conditional random fields. In *Proc. ACL*, 2005.

N. Smith. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. PhD thesis, Johns Hopkins University, 2006.

N. Smith and J. Eisner. Guiding unsupervised grammar induction using contrastive estimation. In *Proc. IJC-AI Workshop: Grammatical Inference Applications*, 2005a.

N. Smith and J. Eisner. Annealing structural bias in multilingual weighted grammar induction. In *Proc. ACL*, 2006.

N. A. Smith and J. Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. ACL*, pages 354–362, 2005b.

B. Snyder and R. Barzilay. Unsupervised multilingual learning for morphological segmentation. In *Proc. ACL*, 2008.

B. Snyder, T. Naseem, and R. Barzilay. Unsupervised multilingual grammar induction. In *Proc. ACL-IJCNLP*, 2009a.

B. Snyder, T. Naseem, J. Eisenstein, and R. Barzilay. Adding more languages improves unsupervised multilingual part-of-speech tagging: a bayesian non-parametric approach. In *Proc. NAACL*, 2009b.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. From Baby Steps to Leapfrog: How "Less is More" in unsupervised dependency parsing. In *Proc. of NAACL-HLT*, 2010.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

J. Tiedemann. Building a multilingual parallel subtitle corpus. In *Proc. CLIN*, 2007.

K. Toutanova, D. Klein, C. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. HLT-NAACL*, 2003.

P. Tseng. An analysis of the EM algorithm and entropy-like proximal point methods. *Mathematics of Operations Research*, 29(1):27–44, 2004.

Y. Tsuruoka and J. Tsujii. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proc. HLT-EMNLP*, 2005.

L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

L. Van der Beek, G. Bouma, R. Malouf, and G. Van Noord. The Alpino dependency treebank. *Language and Computers*, 2002.

S. Vogel, H. Ney, and C. Tillmann. Hmm-based word alignment in statistical translation. In *Proc. COLING*, 1996.

H. Yamada and Y. Matsumoto. Statistical dependency analysis with support vector machines. In *Proc. IWPT*, 2003.

D. Yarowsky and G. Ngai. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proc. NAACL*, 2001.