Learning Large-Scale Conditional Random Fields

Joseph K. Bradley

January 2013
CMU-ML-13-100

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Carlos Guestrin (U. of Washington, Chair)
Tom Mitchell
John Lafferty (U. of Chicago)
Andrew McCallum (U. of Massachusetts at Amherst)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

This research was sponsored by the National Science Foundation under grant numbers IIS0644225, CNS0625518, CNS0721591, the Office of Naval Research under grant number N000140710747, the US Army under grant number W91F0810242, and by fellowships from the National Science Foundation and from the Intel Science and Technology Center.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Abstract

Conditional Random Fields (CRFs) [Lafferty et al., 2001] can offer computational and statistical advantages over generative models, yet traditional CRF parameter and structure learning methods are often too expensive to scale up to large problems. This thesis develops methods capable of learning CRFs for much larger problems. We do so by decomposing learning problems into smaller, simpler subproblems. These decompositions allow us to trade off sample complexity, computational complexity, and potential for parallelization, and we can often optimize these trade-offs in model- or data-specific ways. The resulting methods are theoretically motivated, are often accompanied by strong guarantees, and are effective and highly scalable in practice.

In the first part of our work, we develop core methods for CRF parameter and structure learning. For parameter learning, we analyze several methods and produce PAC learnability results for certain classes of CRFs. Structured composite likelihood estimation proves particularly successful in both theory and practice, and our results offer guidance for optimizing estimator structure. For structure learning, we develop a maximum-weight spanning tree-based method which outperforms other methods for recovering tree CRFs. In the second part of our work, we take advantage of the growing availability of parallel platforms to speed up regression, a key component of our CRF learning methods. Our `Shotgun` algorithm for parallel regression can achieve near-linear speedups, and extensive experiments show it to be one of the fastest methods for sparse regression.

Acknowledgments

I would first like to thank my advisor Carlos Guestrin and his research group. Carlos' advice—on both specific research topics and more generally on how to succeed in research—has been invaluable. Moreover, he has put together an amazing group of students who help each other in research, presentation, writing, and life. I would especially like to thank Aapo Kyrola and Danny Bickson for our very productive collaboration on parallel regression, as well as Stanislav Funiak for his initial leadership in developing the codebase I used for my research.

I would also like to thank previous advisors: Eric Xing, who oversaw my first year as a graduate student, Ali Rahimi, who guided me through my first research internship, and Moises Goldszmidt, who offered me excellent advice on research and my future. I would particularly like to thank Rob Schapire, with whom I did my first major research project and who inspires lasting loyalty in all students whom he teaches.

With respect to specific projects, I am grateful to John Lafferty and Geoff Gordon for feedback on my composite likelihood work. Many thanks to Tom Mitchell, Mark Palatucci, and Dean Pomerleau for helping me with the fMRI application and providing their data. I also thank John Langford, Guy Blelloch, Joseph Gonzalez, and Yucheng Low for helpful feedback on our parallel sparse regression work. I appreciate the time Tom Mitchell, Bryan Kisiel, and Andrew Carlson took to discuss NELL and provide access to their data. I would like to thank my thesis committee for feedback and advice on all of these projects.

Finally, I could not have made it through grad school without my family and friends. I am very grateful to my parents for their support, to my brother Rob for his always-excellent advice and attitude, and to Liza for keeping me in my place. Thank you, Frances, for your patience and support during these stressful days. And thanks to the CMU Ballroom Dance Club members for providing a haven from computers and a place for art, exercise, and laughter.

Contents

Chapter 1

Introduction

Probabilistic Graphical Models (PGMs) are models of probability distributions whose graphical structure encodes independence assumptions among random variables. (See, e.g., Koller and Friedman [2009].) Techniques for learning the parameters and structure of PGMs (e.g., Chow and Liu [1968], Lafferty et al. [2001]), as well as applications of PGMs to real-world problems (e.g., Schmidt et al. [2008], Vail et al. [2007]), have progressed rapidly. PGMs permit modeling complex interactions between random variables, are often intuitive and interpretable models, and provide principled frameworks for probabilistic learning and inference.

Conditional Random Fields (CRFs), introduced by Lafferty et al. [2001], model conditional distributions $P(\mathcal{Y}|\mathcal{X})$. CRFs offer computational and statistical advantages over analogous generative models of the joint distribution $P(\mathcal{Y}, \mathcal{X})$. However, current parameter and structure learning methods are often too expensive to scale to large problems, thus limiting their general utility.

In this thesis, we develop methods for scaling CRF learning to very large problems. In the first part of this work, we develop efficient algorithms for parameter learning (Ch. 2) and structure learning (Ch. 3). The second part of this work (Ch. 4) applies parallel computation to regression, which is an integral part of our CRF parameter and structure learning methods.

Throughout our work, we use methods which are both motivated by theory—achieving strong theoretical guarantees in many cases—and practical—demonstrating their efficacy and scalability on large problems. We emphasize two major themes. The first is *decomposition*: our methods decompose difficult learning problems into simpler parts by identifying locality in the computation. The second is *trade-offs*: most of our methods trade off two or more of sample complexity, computational complexity, and potential for parallelization, where a small sacrifice in one permits a large gain in another.

Thesis Statement: Structured, discriminative probabilistic models offer computational and statistical advantages over more commonly used models, but traditional learning methods are often impractical for large problems. We can scale learning to much larger applications by using decompositions of learning problems which trade off sample complexity, computation, and parallelization.

1.1 Probabilistic Graphical Models

We briefly describe *Probabilistic Graphical Models (PGMs)*. For more details, we refer the reader to, e.g., Koller and Friedman [2009]. A PGM represents a probability distribution over a set of *random variables* via a graphical *structure* and an accompanying *parameterization*. The structure provides an interpretable encoding of independence relations between the random variables, and it provides a framework for computation on the probability distribution. Parameters are associated with specific parts of the graphical structure, and their combination encodes the actual probability values. This combination of interpretability, flexibility, and sound statistical methodology provided by the PGM framework has allowed countless successful applications, from natural language processing tasks such as part of speech tagging [Lafferty et al., 2001] to activity recognition [Vail et al., 2007] and heart motion abnormality detection [Schmidt et al., 2008].

Within the class of PGMs, several popular subclasses include Bayesian networks [Pearl, 1985], Markov Random Fields (MRFs) [Kindermann and Snell, 1980], and Conditional Random Fields (CRFs) [Lafferty et al., 2001]. Bayesian networks are *directed* graphical models, while MRFs and CRFs are *undirected*. In this thesis, we work with undirected models.

We describe MRFs and CRFs in the next subsections. Readers familiar with MRFs and CRFs may skip these subsections, but we remind those readers that *inference* poses a major computational barrier for general models. The expense of inference—and the importance of inference for traditional learning methods—motivates our discussion of alternative learning methods, resumed in Sec. 1.2.

1.1.1 Markov Random Fields (MRFs)

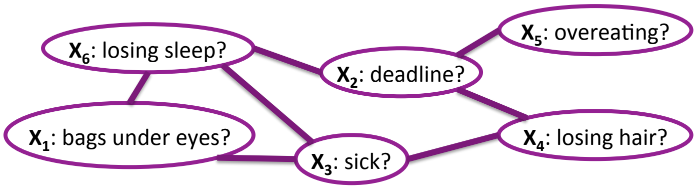
We describe MRFs, explaining terminology and notation for PGMs. MRFs are *generative* models, meaning that they encode non-conditional probability distributions $P(\mathcal{X})$, where $\mathcal{X}$ is a set of random variables. We denote individual random variables as $X_i \in \mathcal{X}$ and sets of variables as $X_{C_i} \subseteq \mathcal{X}$, where C_i is a set of indices. Uppercase X_i denotes a random variable, while lowercase x_i denotes a specific value of X; we write $x_i \in Val(X_i)$. The two key concepts in defining an MRF are the graphical *structure* and the *parameterization*.

We show an example graphical structure for an MRF in Fig. 1.1. Each node in the graph corresponds to a random variable $X_i \in \mathcal{X}$. Each edge in the graph indicates a direct interaction between the two endpoint variables. A random variable is *conditionally independent* of all other variables in the model, given values for its *neighbors*; this property is called local Markov independence. A variable's neighbors are referred to as the variable's *Markov blanket*. In the simple model in Fig. 1.1, if we observe whether the student has been losing sleep (X_6), then the probability that the student has an upcoming paper deadline (X_2) will not change if we check whether the student has bags under his eyes (X_1). We denote the set of neighbors of variable X_i as X_{N_i} and the neighbors of a set of variables X_C as X_{N_C}.

The graphical structure of an MRF corresponds to its parameterization. An MRF may be written as a product of *factors* ψ_j:

$$P(\mathcal{X}) = \frac{1}{Z} \prod_j \psi_j(X_{C_j}), \tag{1.2}$$

where each factor is a function $\psi_j : Val(X_{C_j}) \rightarrow \mathbb{R}_+$ (where $\mathbb{R}_+$ denotes non-negative real values). We discuss specific parameterizations of ψ_j in later chapters. X_{C_j} is called the *domain* of factor ψ_j, and each



$$P(\mathcal{X}) = \frac{1}{Z} \cdot \psi_{1,6}(X_1, X_6) \cdot \psi_{1,3}(X_1, X_3) \cdot \psi_{2,6}(X_2, X_6) \cdot \psi_{3,6}(X_3, X_6) \cdot \psi_{2,5}(X_2, X_5) \cdot \psi_{2,4}(X_2, X_4) \cdot \psi_{3,4}(X_3, X_4) \tag{1.1}$$

Figure 1.1: **Example MRF: Modeling a graduate student's health.** *Random variables (nodes)* X_i represent real-world measurements (boolean-valued in this model). An *edge* between two variables X_i, X_j indicates a direct interaction, i.e., the presence of a *factor* $\psi(X_i, X_j)$ in the model. The graphical *structure* of the model defines statistical assumptions and the computation needed for inference and learning.

$X_i \in X_{C_j}$ is called an *argument* of the factor. We say that X_i *participates* in factor ψ_j. Z is called the *partition function* (a.k.a. *normalization constant*), and it is chosen s.t. $P(\mathcal{X})$ is a probability distribution which sums to 1:

$$Z = \sum_{x \in Val(\mathcal{X})} \prod_j \psi_j(X_{C_j}). \tag{1.3}$$

The factor domains X_{C_j} determine the graphical structure of an MRF: a variable X_i shares an edge with a variable X_k if and only if $i, k \in C_j$ for some j. In Fig. 1.1, factor $psi_{1,6}$ with domain $\{X_1, X_6\}$ corresponds to the edge between X_1 and X_6 in the graphical structure. The MRF in Fig. 1.1 has only binary factors (with two arguments per factor). A factor with more than two arguments corresponds to a hyperedge in the graphical structure.

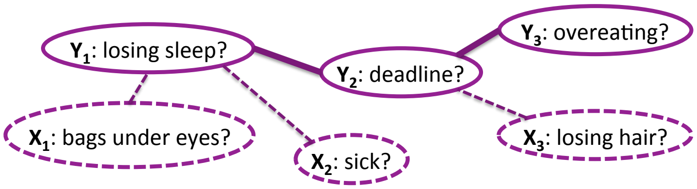
1.1.2 Inference

Given an MRF's factors $\{\psi_j\}$, we must perform *inference* in order to compute probabilities such as $P(\mathcal{X})$, $P(X_A)$, or $P(X_A|x_B)$. For $P(\mathcal{X})$, inference means computing the partition function Z. For $P(X_A)$, inference means *summing out* all variables not in X_A (summing over the values of all variables $\mathcal{X} \setminus X_A$). For $P(X_A|x_B)$, we must *condition* our model on the values x_B: we instantiate $X_B = x_B$ in our model, sum out all variables not in $X_A \cup X_B$, and then compute the partition function for the resulting model over X_A.

In general, exact inference in MRFs is #P-hard, and approximate inference is NP-hard [Roth, 1996]. However, for certain classes of MRFs, inference can be done efficiently. Perhaps the most popular class of MRFs which permit tractable inference are *low-treewidth* models. Intuitively, the treewidth of a model measures how tree-like the structure is: a tree has treewidth 1 and permits efficient inference, while a fully connected structure has treewidth $|\mathcal{X}| - 1$ and does not permit tractable inference.

1.1.3 Conditional Random Fields (CRFs)

This thesis focuses on *Conditional Random Fields (CRFs)* [Lafferty et al., 2001], which generalize MRFs. CRFs model conditional probability distributions $P(\mathcal{Y}|\mathcal{X})$. We call $\mathcal{Y}$ the set of *output variables* and $\mathcal{X}$ the set of *input variables*.



$$P(\mathcal{X}) = \frac{1}{Z(\mathcal{X})} \cdot \psi(Y_1, X_1) \cdot \psi(Y_1, X_2) \cdot \psi(Y_1, Y_2) \cdot \psi(Y_2, X_3) \cdot \psi(Y_2, Y_3) \tag{1.5}$$

Figure 1.2: **Example CRF: Modeling a graduate student's health.** Compare with the MRF in Fig. 1.1: in this CRF, the output variables $\mathcal{Y}$ and input variables $\mathcal{X}$ together form the variables in the MRF. Conditioning on $\mathcal{X}$ removes the structure over $\mathcal{X}$ from the model, simplifying the representation.

Like MRFs, CRFs may be written as a product of factors:

$$P(\mathcal{Y}|\mathcal{X}) = \frac{1}{Z(\mathcal{X})} \prod_i \psi_j(Y_{C_j}, X_{D_j}), \tag{1.4}$$

where a factor ψ_j can be a function of both output and input variables. For CRFs, we use C_j to index domains of output variables and D_j to index domains of input variables. Note that the CRF partition function $Z(\mathcal{X})$ is a function of $\mathcal{X}$. Just as we had to recompute Z for MRFs $P(\mathcal{X})$ after conditioning on a subset of $\mathcal{X}$, we must compute $Z(x)$ for CRFs in order to compute $P(\mathcal{Y}|\mathcal{X} = x)$.

The graphical structure for CRFs is defined w.r.t. only the output variables. I.e., the statistical assumptions implicit in a CRF are made primarily w.r.t. the output variables, and fewer assumptions need to be made about the input variables. One way to understand this benefit is as follows: Given an MRF for $P(\mathcal{Y}, \mathcal{X})$ (the *joint* distribution) written as a product of factors, we can express a CRF for $P(\mathcal{Y}|\mathcal{X})$ as a product of the same factors, excluding those factors which do not have $\mathcal{Y}$ variables in their domains. Thus, CRFs avoid modeling a distribution over $\mathcal{X}$. Moreover, the graphical structure is simplified to be over $\mathcal{Y}$ only, not $\mathcal{X}$. Even if $P(\mathcal{Y}, \mathcal{X})$ is high-treewidth, the conditional $P(\mathcal{Y}|\mathcal{X})$ may not be. (If $P(\mathcal{Y}, \mathcal{X})$ is low-treewidth, so is $P(\mathcal{Y}|\mathcal{X})$.) We give an example of a CRF in Fig. 1.2, in which $\mathcal{Y} \cup \mathcal{X}$ are the variables in the example MRF in Fig. 1.1.

Since CRFs generalize MRFs, the hardness results for MRF inference apply to CRFs as well. However, a CRF modeling $P(\mathcal{Y}|\mathcal{X})$ can permit tractable inference when a corresponding MRF modeling $P(\mathcal{Y}, \mathcal{X})$ requires intractable inference, for the graphical structure of the CRF could be much simpler than that of the MRF. On the other hand, the dependence of the partition function $Z(\mathcal{X})$ on the values of $\mathcal{X}$ can make traditional parameter learning for CRFs very expensive, as discussed in Sec. 1.2.

1.2 Parameter Learning

Parameter learning refers to choosing the values of the factors $\psi_j(Y_{C_j}, X_{D_j})$. Perhaps the most traditional learning method is maximum-likelihood estimation (MLE), which optimizes the (conditional) log likelihood of the data. Optimizing via an iterative gradient-based algorithm requires running inference on every iteration in order to compute the objective and/or its gradient must be computed. Thus, this learning approach is tractable for low-treewidth structures (e.g., Lafferty et al. [2001]) but not for general models.

CRFs additionally complicate learning since inference must be run separately for every training example (since the partition function depends on $\mathcal{X}$).

In Ch. 2, we approach parameter learning with the goal of PAC-style learning for high-treewidth models. The *Probably Approximately Correct (PAC)* framework, proposed by Valiant [1984], sets a standard for learnability guarantees: a model is PAC-learnable if there exists an algorithm which achieves high accuracy (approximately correct) with high probability, with the additional constraint of using a polynomial amount of computation (and thus polynomially many examples).

Most previous applications of learning MRFs and CRFs on high-treewidth models used either approximate objectives (rather than the MLE objective) or approximate inference to make learning tractable. Some empirically successful approximations include pseudolikelihood [Besag, 1975], composite likelihood [Lindsay, 1988], piecewise likelihood [Sutton and McCallum, 2005], and contrastive divergence [Hinton, 2002]. Unfortunately, most such approximations did not come with PAC-style guarantees on the quality of the resulting model, though some methods worked well empirically.

To our knowledge, before our work, the only previous work proving PAC-style learnability for any class of high-treewidth models was the *canonical parameterization* of Abbeel et al. [2006]. Their method reparameterizes the model into a set of local factors which may be estimated separately, without intractable inference. In Sec. 2.6, we show how to extend this method from MRFs to CRFs and derive several improvements to the method. However, we also demonstrate that the canonical parameterization essentially reduces to pseudolikelihood, which often performs better empirically. Nevertheless, we argue that the canonical parameterization may yet prove useful for theoretical analysis, even if it is not the most practical algorithm.

In Ch. 2, we primarily discuss pseudolikelihood and composite likelihood, which decompose the learning problem (over all variables) into smaller problems (over small sets of variables). Before our work, these methods were often labeled as heuristics since they lacked finite-sample quality guarantees. We are able to prove that these methods allow PAC-style learnability for an even more general class of models than the canonical parameterization allows. Our theoretical results reveal how pseudolikelihood makes learning tractable by trading off slightly larger sample complexity for much lower computational complexity, with the side benefit of much easier parallelization. Composite likelihood, which generalizes pseudolikelihood and the full likelihood (MLE) and many intermediate objectives, opens up a wide range of possible trade-offs; most importantly, *we are able to tailor composite likelihood to the model and the dataset to optimize the trade-off* between sample complexity, computational complexity, and potential for parallelism. This *structured composite likelihood* approach reveals many new avenues for research.

1.3 Structure Learning

Our work on parameter learning assumes that the structure of the model is pre-specified. *Structure learning* refers to selecting outputs Y_{C_j} for each factor, thus choosing the graphical structure over $\mathcal{Y}$ representing conditional independence relations in the distribution. We distinguish structure learning from *feature selection*, which refers to selecting inputs X_{D_j} for each factor, i.e., inputs directly relevant to Y_{C_j}.

Significant work has been done on structure learning for generative models. While Srebro [2003] showed that MRF structure learning is NP-hard in general, some methods can provably learn certain classes of MRFs. Tree-structured MRFs may be learned using the famous Chow-Liu algorithm [Chow and Liu, 1968]. More general low-treewidth models may be learned using the constraint-based method proposed by Chechetka and Guestrin [2007]. Ravikumar et al. [2008, 2010] use L_1 regularization to recover sparse structures, with theoretical guarantees for some classes of models. In addition, heuristic methods such as local search (e.g., Teyssier and Koller [2005]) can perform well in practice.

However, few papers address CRF structure learning. Moreover, although expert knowledge can sometimes dictate structure and features, conditional independence structure in CRFs can be much less intuitive than the independence structure in MRFs. There are few learnability results for CRF structure. Learning is known to be hard in general; this result follows from the hardness results for MRFs from Srebro [2003]. The results on learning Ising MRF structures from Ravikumar et al. [2010] imply learnability for some classes of Ising CRFs. Nevertheless, several papers have achieved empirical success using heuristic methods, such as greedy factor selection [Torralba et al., 2004], block-ℓ_1-regularized pseudolikelihood [Schmidt et al., 2008], and approximate objectives [Shahaf et al., 2009].

In Ch. 3, we discuss methods for learning tree structures, which are particularly useful since they permit tractable exact inference. Given the success of the Chow-Liu algorithm in learning tree MRFs, we use the same algorithmic approach for CRFs: weight each possible edge, and choose a maximum-weight spanning tree. The approach is very computationally efficient, and the decomposition into disjoint computation of edge weights permits simple parallelization. Unfortunately, the obvious generalization of the Chow-Liu algorithm to CRFs quickly breaks down as the number of input variables grows, for the edge weights, conditional mutual information (CMI), become difficult to estimate.

In Ch. 3, we propose a class of edge weights generalizing CMI. Although we can prove a negative result showing that no single member of the class of edge weights is sufficient for recovering all tree CRFs, we demonstrate that two members of the class significantly outperform CMI in practice. In Sec. 3.6, we outline potential algorithmic improvements which might sidestep our negative result, as well as ideas for performing feature selection jointly with structure learning.

1.4 Parallel Regression

So far, we have introduced our work on core learning methods for CRF parameters and structure. We now discuss work on scaling a core component of those methods: *regression*, or estimating one variable (or a small set of variables) given the values of other variables (called *covariates*). In parameter learning, regression is a key step in pseudolikelihood (Ch. 2) and in the canonical parameterization (Sec. 2.6). In structure learning, regression is used to estimate edge weights in our method (Ch. 3) and in many neighborhood estimation methods (e.g., Ravikumar et al. [2008, 2010]). Our work scales regression using parallel computation.

For many years, people have relied on exponential growth in processor speeds to provide ever-increasing scalability for their algorithms. Recently, processor speeds have essentially stopped increasing, and scalability via hardware must instead come from taking advantage of parallel computing. Multicore and distributed computing systems have become widely available, and interest in parallel algorithms—and increasingly in parallel machine learning—has followed suit.

In Ch. 4, we examine parallel optimization for sparse (L_1-regularized) regression. Sparse regression is of particular interest since sparsity-inducing regularization allows regression methods to scale well w.r.t. the number of irrelevant covariates [Ng, 2004]. As we argue, many methods with obvious parallelization schemes are relatively inefficient for sparse regression.

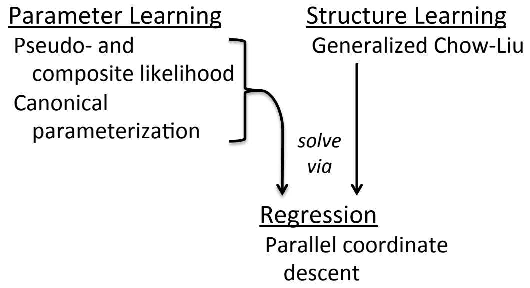

Parameter Learning
Pseudo- and composite likelihood
Canonical parameterization
Structure Learning
Generalized Chow-Liu
solve via
Regression
Parallel coordinate descent

Figure 1.3: **Thesis overview: CRF learning problems and our solutions.** Learning CRFs requires learning parameters and structure. For parameters, we study three methods: pseudolikelihood, composite likelihood, and the canonical parameterization. For structure, our method is a generalized Chow-Liu algorithm [Chow and Liu, 1968]. All of our parameter and structure learning methods use regression as a key component. For regression, our method is parallel coordinate descent.

We examine coordinate descent, which optimizes one parameter while holding all others fixed. Coordinate descent is known to be one of the fastest existing methods for sparse regression, yet it seems inherently sequential at first glance. We prove the opposite: a simple parallel version of coordinate descent, dubbed `Shotgun` [Bradley et al., 2011], achieves near-linear speedups up to a data-dependent limit. Moreover, we demonstrate the algorithm's practicality through extensive empirical tests which show `Shotgun` to be one of the fastest and most robust methods for sparse regression.

`Shotgun`'s success is due to data-specific locality in the optimization problem which limits interactions between coordinate updates. This locality allows the algorithm to decompose the update of a set of variables into independent updates. While this simplification can lead to slightly higher total computational cost, it permits much greater parallelism. While our current work is for multicore systems, we propose extensions of our ideas to distributed systems, in which communication becomes a major bottleneck.

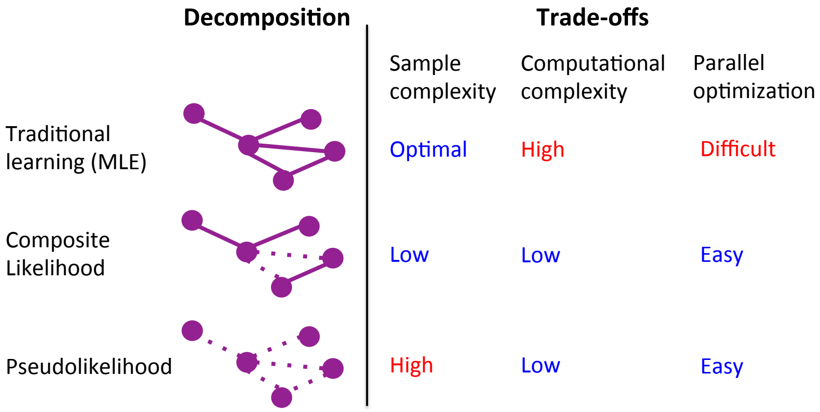
1.5 Summary and Main Contributions

We study three problems: CRF parameter learning, CRF structure learning, and regression. We detail these problems and our solutions in Fig. 1.3, showing how they relate. Our solutions are guided by the main themes of this thesis: *decomposing* problems and optimizing *trade-offs* to make learning more scalable. We showcase our work on CRF parameter learning in Fig. 1.4 to illustrate these themes.

1.5.1 Contributions

We summarize our main contributions below and in Fig. 1.5. For each part of CRF learning—parameter learning, structure learning, and regression—we make contributions to both theory and practice.

Parameter learning: We prove finite sample complexity bounds for learning the parameters of general MRFs and CRFs by using MLE, pseudolikelihood, and composite likelihood. Our bounds imply PAC-style learnability of certain general classes of MRFs and CRFs, and they provide theoretical justification for methods previously considered to be heuristics. We also improve the canonical parameterization of Abbeel et al. [2006] and relate it to pseudolikelihood. Our analysis gives practical guidance for choosing



Decomposition	Sample complexity	Computational complexity	Parallel optimization
Traditional learning (MLE)	Optimal	High	Difficult
Composite Likelihood	Low	Low	Easy
Pseudolikelihood	High	Low	Easy

Figure 1.4: **Thesis themes: decomposition and trade-offs.** Two of our parameter learning methods, pseudolikelihood and composite likelihood, showcase the themes in this thesis. We *decompose* the problem of parameter learning by splitting the optimization over the full model (in MLE) into a set of smaller sub-problems. This allows us to trade some sample complexity for improved computational complexity and parallelism. Composite likelihood lets us optimize this trade-off w.r.t. our model and data.

	Theory	Practice
Parameter Learning	Sample complexity bounds; Improved canonical parameterization	Guidance for choosing problem-specific composite likelihood estimators
Structure Learning	Analysis of issues in generalizing Chow-Liu	Scalable method for learning tree CRFs
Parallel Regression	Proof of near-linear speedups for parallel coordinate descent	Shotgun, one of the fastest multicore algorithms for sparse regression

Figure 1.5: **Main contributions in this thesis**

problem-specific estimators to optimize trade-offs between sample complexity, computation, and parallelism.

Structure learning: We study generalizing the Chow-Liu algorithm [Chow and Liu, 1968] to CRFs and provide a deeper understanding of problems which arise. We propose a scalable heuristic method for learning tree structures for CRFs which works well in practice.

Parallel regression: We prove convergence bounds for a simple parallel version of coordinate descent, showing that it can achieve near-linear speedups up to a problem-dependent limit. In extensive experiments, we demonstrate our `Shotgun` algorithm to be one of the fastest multicore algorithms for sparse regression.

Chapter 2

CRF Parameter Learning

As discussed in Sec. 1.2, traditional parameter learning via *maximum-likelihood estimation (MLE)* for *Markov Randoms Fields (MRFs)* and *Conditional Random Fields (CRFs)* is very expensive due to the cost of running *inference*. For high treewidth models, inference is intractable, so MLE is intractable as well. When approximate inference is used, MLE generally loses its statistical guarantees. Some methods, such as *maximum pseudolikelihood estimation (MPLE)* [Besag, 1975], estimate parameters without intractable inference (i.e., using only tractable probabilistic queries) but—before our work—only had asymptotic guarantees for general models [Liang and Jordan, 2008].

In this section, we discuss scaling up parameter learning to much larger problems using *structured maximum composite likelihood estimation (MCLE)*. MCLE generalizes both MLE and MPLE and allows a much more flexible choice of estimator structure. MCLE was proposed by Lindsay [1988]. Our primary contributions to the method are an improved analysis and a better understanding of estimator structure, which permits more effective choices of composite likelihood estimators in practice.

Our analysis of MCLE gives the first strong finite sample guarantees for learning the parameters of general CRFs (which generalize MRFs). We prove learning bounds for MCLE w.r.t. the parameter estimation error and log loss when the target distribution is in our model class. In the large sample limit, our bounds match existing asymptotic normality results for MCLE [Liang and Jordan, 2008].

For certain classes of models, our bounds fall within the probably approximately correct (PAC) framework [Valiant, 1984]: we can achieve high accuracy with polynomial sample and computational complexity. As we will see, a problem-dependent factor in our bounds determines whether the sample complexity of learning grows slowly as model size increases (permitting PAC learning) or grows quickly (preventing PAC learning).

We include a detailed empirical analysis of our theoretical results. We show that our bounds accurately capture MCLE behavior in terms of a few problem properties, and we study how those properties vary with model structure and parameters. Finally, we improve upon the traditional use of MCLE by showing that careful choice of the MCLE loss structure can provide computationally efficient estimators with better statistical properties and empirical performance.

To our knowledge, only one other method for learning parameters of high-treewidth discrete models has PAC guarantees: the *canonical parameterization* of Abbeel et al. [2006], which re-factorizes the model into factors which may be estimated separately. In Sec. 2.6, we present a detailed analysis of their method, deriving improvements both to the learning algorithm and its bounds. However, in Sec. 2.6.5, we argue that their method is very similar to MPLE and that MPLE (and MCLE) should perform better in practice.

Our main contributions in this chapter may be summarized as follows:

- Finite sample complexity bounds for general MRFs and CRFs, using MPLE and MCLE
    - PAC learnability for certain classes of high-treewidth models
- Empirical analysis of bounds on various models, comparing MLE, MPLE, and MCLE
- Guidelines for choosing MCLE structures to optimize trade-offs between sample complexity, computation, and parallelism
- Analysis of the canonical parameterization of Abbeel et al. [2006]
    - Improvements to the algorithm and bounds
    - Better understanding of the relation with MPLE

Our work on composite likelihood was initially presented in Bradley and Guestrin [2012].

2.1 Related Work

We first discuss general categories of parameter learning methods to frame our approaches. We then discuss work closely related to our work on MPLE and MCLE. Finally, we mention the few pieces of work related to the the canonical parameterization.

2.1.1 Categories of Parameter Learning Methods

We mention several broad categories of parameter learning methods. As discussed in Sec. 1.2, traditional *MLE* training (e.g., Lafferty et al. [2001]) is optimally statistically efficient but computationally intractable for many models. *Batch MLE* training may be sped up by using *stochastic gradient* estimates, where the gradient is estimated using a single random training example or a mini-batch [Vishwanathan et al., 2006]. However, the use of stochastic gradient estimates does not affect the intractability of inference.

Many methods avoid intractable inference by modifying the likelihood objective. One such class of methods compares ratios of probabilities of subsets of variables in the model; in these ratios, the two copies of the partition function cancel out, eliminating the need for intractable inference. *Pseudolikelihood* [Besag, 1975] and *composite likelihood* [Lindsay, 1988], which we analyze in this chapter, belong to this class. Before our work, these two methods were known to be consistent (e.g., Liang and Jordan [2008]) and empirically useful (e.g., Schmidt et al. [2008], Sutton and McCallum [2005, 2007]). We are able to prove that both methods come with finite sample guarantees for many models as well. Another member of this class is *ratio matching* [Hyvärinen, 2007], which is consistent but not always convex. (Pseudolikelihood and composite likelihood objectives are convex.)

Several other alternative methods for modifying the likelihood objective exist. Piecewise likelihood and piecewise pseudolikelihood [Sutton and McCallum, 2005, 2007], which train each factor separately, are not consistent for all models, but they have been shown to work well in practice in the domain of natural language processing. Score matching [Hyvärinen, 2005] is consistent with perfect numerical optimization, but the objective is not always convex.

Another class of methods replace intractable inference with approximate inference which does not come with an a priori quality guarantee. Common methods for approximate inference include Gibbs sampling [Geman and Geman, 1984], which lacks finite-time guarantees on the approximation accuracy of inference results, and variational inference (e.g., Wainwright [2006]), which sometimes includes problem-specific guarantees computable at runtime. Hybrid methods also exist: contrastive divergence [Hinton, 2002] combines Gibbs sampling with stochastic gradient estimates and an alternative objective function.

The (tractable) methods listed above are often empirically successful but, to our knowledge, do not come with PAC-style learning guarantees for general MRFs and CRFs. Before our work, the one exception was the *canonical parameterization* of Abbeel et al. [2006], which we discuss more in Sec. 2.6. Abbeel et al. [2006] used this method to prove PAC-style bounds for MRFs with discrete variables representable as bounded-degree factor graphs. In Sec. 2.6, we generalize their work to CRFs and derive several improvements. Yet we also show that their algorithm is very similar to pseudolikelihood and is arguably less practical, though perhaps still useful for theoretical analysis.

2.1.2 Pseudolikelihood and Composite Likelihood

Several authors have compared MLE with MPLE and MCLE. Many works have shown MPLE to be empirically successful (c.f., Schmidt et al. [2008], Sutton and McCallum [2005, 2007]), achieving lower accuracy but requiring less computation than MLE. Theoretical analyses have predicted such behavior (c.f., Gidas [1986], Hyvärinen [2006], Liang and Jordan [2008]), but only in the asymptotic setting.

Much of this work is on *structured MCLE*, which uses composite likelihood components which are structured according to the model being learned and can therefore be quite large while still maintaining tractable inference. Previous work generally used small components, often not chosen according to the model structure. For example, Dillon and Lebanon [2010] tested MCLE components composed of all subsets of two to four variables, which gave better empirical performance than MPLE. One notable exception was Asuncion et al. [2010], who discussed the benefits of using of small tree-structured components for MCLE (though they used the components for sampling-based inference, not a deterministic estimator).

MCLE has also been generalized to *stochastic composite likelihood* [Dillon and Lebanon, 2010], which stochastically combines computationally cheap estimators such as MPLE with expensive ones such as MLE. Rather than using the same estimator for each training example, an estimator is selected randomly, thus decreasing the number of times expensive estimators are used. This idea could be readily combined with our ideas for structured estimators.

Part of our discussion involves handling inconsistent parameter estimates from disjoint pseudolikelihood optimization (Sec. 2.3.4). Lowd [2012] discuss similar issues in the context of translating dependency networks to MRF representations. Dependency networks represent a distribution as a set of conditional distributions for each variable: $P(X_i|X_{N_i})$; thus, learning a dependency network is identical to pseudolikelihood with disjoint optimization. Lowd [2012] present an in-depth discussion of handling inconsistent estimates using averaging schemes, with an emphasis on models over Boolean variables.

Our analysis is most closely related to that of Ravikumar et al. [2010], who proved sample complexity bounds for parameter estimation error in regression problems $Y_i \sim X$ in Ising models. We adapt their analysis to handle general log linear CRFs, structured losses, and shared parameters (Sec. 2.3.4), and we use the resulting bounds on parameter estimation error to prove bounds on the log loss. Our bounds resemble asymptotic normality results such as those of Liang and Jordan [2008] and Dillon and Lebanon [2010].

2.1.3 Canonical Parameterization

To our knowledge, the canonical parametrization initially proposed by Abbeel et al. [2006] has not been studied by many later works. The one exception was Roy et al. [2009], who derived a significant improvement which we discuss in Sec. 2.6.1. Their improvement was key in our observations on how the canonical parameterization relates to MPLE.

2.2 Learning CRF Parameters

We define notation here, though a few equations are replicated in Sec. 1.2. We write general log-linear CRFs in the form

$$P_\theta(Y|X) = \frac{1}{Z(X;\theta)} \exp\left(\theta^T \phi(Y,X)\right), \tag{2.1}$$

where Y and X are sets of *output* and *input* variables, respectively. Y and X may be discrete or real-valued. θ is an r-vector of *parameters*; ϕ is an r-vector of *features* which are functions of Y and X with bounded range; and Z is the *partition function*. Note that an MRF $P(Y)$ would be represented by letting $X = \emptyset$.

The feature vector ϕ implicitly defines the structure of the CRF: each element ϕ_t is a function of some subset of Y, X and defines edges in the CRF graph connecting its arguments. We call each $\theta_t^T \phi_t(Y,X)$ a *factor*.

In *parameter learning*, our goal is to estimate θ from a set of i.i.d. samples $\{y^{(i)}, x^{(i)}\}$ from a *target distribution* $P(X)P_{\theta^*}(Y|X)$, where θ^* are the *target parameters*. Note we assume $P_{\theta^*}(Y|X)$ is in our model family in Eq. (2.1). We learn parameters by minimizing a *loss* $\hat{\ell}$, which is a function of the samples and θ, plus a regularization term:

$$\min_\theta \hat{\ell}(\theta) + \lambda \|\theta\|_p. \tag{2.2}$$

Above, $\lambda \geq 0$ is a regularization parameter, and $p \in \{1,2\}$ specifies the L_1 or L_2 norm. We write $\hat{\ell}$ for the loss computed w.r.t. the n training samples and ℓ for the loss w.r.t. the target distribution.

The choice of loss function defines the estimator. MLE minimizes the *log loss*:

$$\ell_L(\theta) = \mathbf{E}_{P(X)P_{\theta^*}(Y|X)} \left[-\log P_\theta(Y|X)\right]. \tag{2.3}$$

MPLE [Besag, 1975] minimizes the *pseudolikelihood loss*, which is the sum over variables Y_i of their likelihoods conditioned on neighbors in $Y_{\setminus i} \doteq Y \setminus \{Y_i\}$ and in X:

$$\ell_{PL}(\theta) = \mathbf{E}_{P(X)P_{\theta^*}(Y|X)} \left[-\sum_i \log P_\theta(Y_i | Y_{\setminus i}, X) \right]. \tag{2.4}$$

In general, computing the conditional probabilities $P(A|\cdot)$ in these losses takes time exponential in $|A|$. Thus, computing the log loss can take time exponential in $|Y|$, while computing the pseudolikelihood loss takes time linear in $|Y|$.

MCLE [Lindsay, 1988] minimizes the *composite likelihood loss*:

$$\ell_{CL}(\theta) = \mathbf{E}_{P(X)P_{\theta^*}(Y|X)}\left[-\sum_i \log P_\theta(Y_{A_i}|Y_{\backslash A_i}, X)\right], \tag{2.5}$$

where Y_{A_i} is a set of variables defining the i^{th} *likelihood component* and $Y_{\backslash A_i} \doteq Y \setminus Y_{A_i}$. With a single component $Y_{A_i} = Y$, MCLE reduces to MLE; if each component is a single variable $Y_{A_i} = \{Y_i\}$, MCLE reduces to MPLE. MCLE thus permits a range of losses between MLE and MPLE with varying computational complexity. We discuss the choice of likelihood components in Sec. 2.5.

If a feature ϕ_t is a function of $Y_i \in Y_{A_i}$, then we say ϕ_t and θ_t *participate* in component A_i, and we write $\theta_t \in \theta_{A_i}$. Also, some works permit more general components conditioned on $Y_B \subseteq Y_{\backslash A_i}$. We restrict $Y_B = Y_{\backslash A_i}$ for simplicity, and our analysis only requires that the set of components $\mathcal{A} \doteq \{A_i\}$ forms a consistent estimator (i.e., an estimator which recovers the target parameters with probability approaching 1 as the training set size approaches infinity).

2.3 Sample Complexity Bounds

This section presents our main theoretical results: finite sample complexity bounds for learning parameters for general CRFs using MCLE. We give bounds in terms of parameter estimation error, which we then use to bound log loss. The appendix has detailed proofs of all results.

We list some of our sample complexity bounds as "PAC-style" bounds since they take the form of PAC bounds: we can achieve low error with high probability given a certain number of training examples. However, each bound includes a factor which can increase with model size, depending on the model structure. We discuss this issue further in Sec. 2.3.5. Also, the PAC framework requires learning to be polytime. MLE and MCLE may or may not be polytime; as discussed in Sec. 2.5, we restrict ourselves to tree-structured MCLE estimators, which permit polytime inference.

2.3.1 Parameter Estimation: MLE

For comparison, we first give bounds for MLE. Our bounds are written in terms of a few model properties: r (the number of parameters), $\phi_{max} \doteq \max_{t,y,x} \phi_t(y, x)$ (the maximum magnitude of any feature vector element), and C_{min} (a lower bound on $\Lambda_{min}\left(\nabla^2 \ell_L(\theta^*)\right)$, the minimum eigenvalue of the Hessian of the log loss w.r.t. the target distribution).[1] Bounding the eigenvalues away from 0 prevents variable interactions from being deterministic.

Our first theorem is a PAC-style bound for MLE w.r.t. parameter estimation error.

Theorem 2.3.1. ***(MLE PAC-style bound)*** *Assume a lower bound on the minimum eigenvalue of the log loss: $\Lambda_{min}\left(\nabla^2 \ell_L(\theta^*)\right) \geq C_{min} > 0$. Suppose we learn parameters $\hat{\theta}$ by minimizing Eq. (2.2) using the log loss ℓ_L w.r.t. $n > 1$ i.i.d. samples, using regularization $\lambda = \frac{C_{min}^2}{2^6 r^2 \phi_{max}^3} n^{-\xi/2}$, where $\xi \in (0, 1)$. Then $\hat{\theta}$ will be close to the target parameter vector θ^*:*

$$\|\hat{\theta} - \theta^*\|_1 \leq \frac{C_{min}}{4r\phi_{max}^3} n^{-\xi/2} \tag{2.6}$$

[1] If the features are overcomplete, then r is the intrinsic dimensionality of the feature space, computed as the number of non-zero eigenvalues of the log loss Hessian. C_{min} is a lower bound on the non-zero eigenvalues.

with probability at least

$$1 - 2r(r+1)\exp\left(-\frac{C_{min}^4}{2^{13}r^4\phi_{max}^8}n^{1-\xi}\right). \tag{2.7}$$

In Theorem 2.3.1, the constant ξ trades off the convergence rate of the parameters with the probability of success. As $n \longrightarrow \infty$, we may let $\xi \longrightarrow 1$ while keeping the probability of success high; as $\xi \longrightarrow 1$, the convergence rate approaches $n^{-1/2}$, the asymptotic rate [Liang and Jordan, 2008]. The next corollary eliminates ξ by converting the PAC-style bound into a sample complexity bound.

Corollary 2.3.2. ***(MLE sample complexity)*** *Given the assumptions from Theorem 2.3.1, to estimate the parameters within L_1 error ϵ with probability at least $1-\delta$, it suffices to have a training set of size*

$$n \geq \frac{2^9\phi_{max}^2}{C_{min}^2}\frac{1}{(\epsilon/r)^2}\log\frac{2r(r+1)}{\delta}. \tag{2.8}$$

This sample complexity result implies that parameters are easier to learn when the minimum eigenvalue bound C_{min} is large; i.e., estimators with large C_{min} are more *statistically efficient*.[2] Asymptotic results (e.g., Liang and Jordan [2008], Ravikumar et al. [2010]) have related statistical efficiency to the loss' Hessian. The above bound is expressed in terms of ϵ/r, the error for the full parameter vector normalized by the number of parameters r; keeping this normalized parameter error ϵ/r constant, the bound increases only logarithmically with r. Also, note that changing ϕ_{max} essentially rescales parameters.

2.3.2 Parameter Estimation: MCLE

We now present bounds generalized to MCLE. Since MCLE can use multiple likelihood components A_i, our bounds contain additional quantities. M_{max} denotes the maximum number of components in which any feature participates. The bound C_{min} applies to all $A_i \in \mathcal{A}$. We also write ρ_{min} as a lower bound on the sum of minimum eigenvalues for likelihood components (w.r.t. the target distribution) in which any parameter θ_t participates: $\rho_{min} \doteq \min_t \rho_t$, where $\rho_t \leq \sum_{i:\theta_t\in\theta_{A_i}} \Lambda_{min}(\nabla^2[\ell_{CL}(\theta^*)]_{A_i})$. Here, $[\ell_{CL}(\theta^*)]_{A_i}$ denotes the loss term contributed by component A_i.

Intuitively, ρ_{min} generalizes C_{min} from MLE to MCLE. Recall that MLE uses a single likelihood component to estimate θ, and the minimum eigenvalue of the component's Hessian (C_{min}) affects the statistical efficiency of MLE. In MCLE, each parameter may be estimated using multiple likelihood components, and the sum of the minimum eigenvalues for those components (ρ_{min}) affects the statistical efficiency of MCLE.

We can now present our PAC-style bound for MCLE.

Theorem 2.3.3. ***(MCLE PAC-style bound)*** *Assume we use a consistent MCLE estimator ℓ_{CL}. Assume bounds $\min_i \Lambda_{min}\left(\nabla^2[\ell_{CL}(\theta^*)]_{A_i}\right) \geq C_{min} > 0$, and let $\rho_{min} = \min_t \rho_t$. Suppose we learn parameters $\hat{\theta}$ by minimizing Eq. (2.2) using the MCLE loss ℓ_{CL} w.r.t. $n > 1$ i.i.d. samples, using regularization $\lambda = \frac{C_{min}^2}{2^6 r^2 M_{max}\phi_{max}^3}n^{-\xi/2}$, where $\xi \in (0,1)$. Then $\hat{\theta}$ will be close to the target parameter vector θ^*:*

$$\|\hat{\theta} - \theta^*\|_1 \leq \frac{\rho_{min}}{4rM_{max}\phi_{max}^3}n^{-\xi/2} \tag{2.9}$$

with probability at least

$$1 - 2r(|\mathcal{A}|r+1)\exp\left(-\frac{C_{min}^4}{2^{13}r^4M_{max}^4\phi_{max}^8}n^{1-\xi}\right). \tag{2.10}$$

[2] We use "statistical efficiency" w.r.t. finite sample sizes, borrowing the term from asymptotic analysis.

If the number of likelihood components is

$$|\mathcal{A}| \leq \tfrac{1}{2r^2}(2r)^{\left[\frac{2^8 C_{min}^2 M_{max}^2}{\rho_{min}^2}\right]}, \tag{2.11}$$

then Eq. (2.9) holds with probability at least

$$1 - 4r \exp\left(-\tfrac{\rho_{min}^4 n^{1-\xi}}{2^{13} r^4 M_{max}^4 \phi_{max}^8}\right). \tag{2.12}$$

We can see that using multiple likelihood components can worsen the bound by increasing M_{max} and $|\mathcal{A}|$ but can also improve the bound by increasing ρ_{min}. Sec. 2.5 shows how careful choices of components can improve this trade-off. We include the special condition Eq. (2.11) since it permits us to replace C_{min} with ρ_{min} in the probability bound, which will later prove helpful in explaining the behavior of MCLE with overlapping components. Eq. (2.11) is a reasonable requirement in many cases, and it holds for our empirical tests.

The following corollary converts Theorem 2.3.3 into a sample complexity bound.

Corollary 2.3.4. ***(MCLE sample complexity)*** *Given the assumptions from Theorem 2.3.3, to estimate the parameters within L_1 error ϵ with probability at least $1 - \delta$, it suffices to have a training set of size*

$$n \geq \tfrac{2^9 M_{max}^2 \phi_{max}^2 \rho_{min}^2}{C_{min}^4} \tfrac{1}{(\epsilon/r)^2} \log \tfrac{2r(|\mathcal{A}|r+1)}{\delta}. \tag{2.13}$$

If Eq. (2.11) holds, then it suffices to have

$$n \geq \tfrac{2^9 M_{max}^2 \phi_{max}^2}{\rho_{min}^2} \tfrac{1}{(\epsilon/r)^2} \log \tfrac{4r}{\delta}. \tag{2.14}$$

Theorem 2.3.3 generalizes Theorem 2.3.1: with MLE, $\rho_{min} = C_{min}$, $M_{max} = 1$, and $|\mathcal{A}| = 1$. For MPLE, we have $\rho_{min} \geq C_{min}$, $M_{max} = 2$, and $|\mathcal{A}| = |Y|$. Thus, MLE's statistical guarantees are stronger than those for MPLE and MCLE only up to problem-dependent constants. All three estimators' sample complexity bounds have the same dependence (up to log terms) on the problem properties r and ϕ_{max}, the desired error ϵ, and the probability of failure δ.

The estimators mainly differ in their spectral properties. Recall that ρ_{min} generalizes C_{min}. If each parameter θ_t participates in an equal number of MCLE components, then we may replace M_{max}/ρ_{min} with the minimum over parameters θ_t of the average of minimum eigenvalues for components in which θ_t participates: $\overline{\rho}_{min} \doteq \min_t \operatorname{avg}_{i\,:\,\theta_t \in \theta_{A_i}} \Lambda_{min}(\nabla^2[\ell_{CL}(\theta^*)]_{A_i})$. This substitution makes our MCLE sample complexity bound in Eq. (2.14) identical (up to log factors) to our MLE bound in Eq. (2.8), with $\overline{\rho}_{min}$ substituted for C_{min}. I.e., MCLE averages the effects of its various components. We show in Sec. 2.5 how this averaging can mitigate the negative impact of "bad" components (with small eigenvalues).

2.3.3 Loss Reduction

Thus far, we have only given bounds on parameter estimation error. We now show that a bound on parameter error may be used to bound the log loss.

Theorem 2.3.5. ***(Log loss, given parameter error)*** *Let Λ_{max} be the largest eigenvalue of the log loss Hessian at θ^*. If the parameter estimation error is small:*

$$\|\theta - \theta^*\|_1 \le \frac{-\Lambda_{max} + \sqrt{\Lambda_{max}^2 + 4r\phi_{max}^4}}{4\phi_{max}^3}, \tag{2.15}$$

the log loss converges quadratically in the error:

$$\ell_L(\theta) \le \ell_L(\theta^*) + \left(\frac{\Lambda_{max}}{2} + \phi_{max}^2\right) \|\theta - \theta^*\|_1^2. \tag{2.16}$$

Otherwise, the log loss converges linearly in the error:

$$\ell_L(\theta) \le \ell_L(\theta^*) + \phi_{max}\|\theta - \theta^*\|_1. \tag{2.17}$$

This theorem describes two well-known convergence regimes: linear far from the optimum and quadratic close to the optimum. In the large sample limit, our results indicate that the log loss of the MLE and MCLE estimates converge at a rate approaching n^{-1}, matching the asymptotic results of Liang and Jordan [2008]. To see this, let $\xi \longrightarrow 1$ in Theorem 2.3.1 and Theorem 2.3.3, and note that we enter the quadratic regime in Theorem 2.3.5.

Sample complexity bounds for MLE and MCLE w.r.t. log loss may be computed by combining Corollary 2.3.2 and Corollary 2.3.4 with Theorem 2.3.5. For lack of space, we relegate these bounds to the appendix.

2.3.4 Disjoint vs. Joint Optimization

In Sec. 2.3.2, we analyzed MCLE using *joint optimization*; i.e., we jointly minimized all likelihood components' contributions to the loss in Eq. (2.5), and parameters θ_t participating in multiple components were shared during optimization. In this section, we discuss *disjoint optimization*, in which each likelihood component is treated as a separate MLE regression problem over a subset of the variables.

With disjoint optimization, each component A_i produces an estimate of its parameter subvector; denote the estimate of each $\theta_t \in \theta_{A_i}$ by $\hat{\theta}_t^{(A_i)}$. These estimates obey the bounds for MLE in Sec. 2.3.1. We can obtain a global estimate $\hat{\theta}$ of the parameters by averaging these subvectors where they overlap:

$$\hat{\theta}_t = \text{avg}_{i\,:\,\theta_t \in \theta_{A_i}} \hat{\theta}_t^{(A_i)}. \tag{2.18}$$

Disjoint optimization is simple to implement and is *data parallel*, permitting easy scaling via parallel computing (c.f., Eidsvik et al. [under submission]). We now show how to bound the error in $\hat{\theta}$ using the bounds from each component's estimate.

Lemma 2.3.6. ***(Disjoint optimization)*** *Suppose we average the results of disjoint optimizations using likelihood components $\mathcal{A}$, as in Eq. (2.18). If each estimated subvector $\hat{\theta}^{(A_i)}$ has L_1 error at most ϵ, then the full estimate has error $\|\hat{\theta} - \theta^*\|_1 \le |\mathcal{A}|\epsilon$.*

The factor of $|\mathcal{A}|$ appears since the estimation error in each $\hat{\theta}^{(A_i)}$ could be in elements of θ which participate only in likelihood component A_i.

Theorem 2.3.7. ***(Disjoint MCLE sample complexity)***
Suppose we average the results of disjoint optimizations using likelihood components $\mathcal{A}$, as in Eq. (2.18).

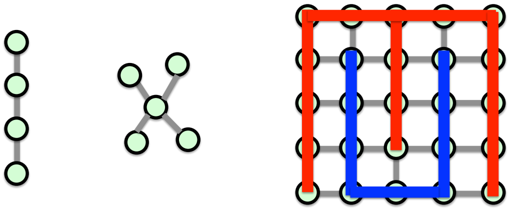


Figure 2.1: **Structures tested.** Left-to-right: chain, star, grid. The grid shows an example of two structured likelihood components ("combs"), as in Sec. 2.5.

Given the assumptions from Theorem 2.3.1 for each component, with a single C_{min} denoting a bound for all components, in order to estimate the parameters θ within L_1 error ϵ with probability at least $1 - \delta$, it suffices to have a training set of size

$$n \geq \frac{2^9 \phi_{max}^2}{C_{min}^2} \frac{|\mathcal{A}|^2}{(\epsilon/r)^2} \log \frac{2r(r+1)|\mathcal{A}|}{\delta}. \tag{2.19}$$

Since $M_{max} \leq |\mathcal{A}|$ and $\rho_{min} \geq C_{min}$, this sample complexity bound for disjoint MCLE is worse than that for joint MCLE in Corollary 2.3.4, as might be expected.

2.3.5 PAC Bounds

We have approached parameter learning with the goal of PAC-style learning. However, the factors C_{min} and ρ_{min} in the above bounds can increase with model size, so our sample complexity bounds are not PAC bounds for all model classes. For some model classes, such as chains and stars with bounded factor strengths, these factors increase at most polynomially with model size, so Theorem 2.3.1 and Theorem 2.3.3 are true PAC bounds. For other model classes, such as models with unbounded factor strengths, these factors can increase super-polynomially with model size, violating the PAC framework's requirements.

2.4 Empirical Analysis of Bounds

We present an extensive study of our sample complexity bounds for MLE and MPLE on a variety of synthetic models. Our results show that our bounds accurately capture the learning methods' behaviors in terms of properties of the target distribution. We show how those properties vary across different model structures and factor types, indicating where MPLE may succeed or fail.

2.4.1 Setup

We tested synthetic models over binary variables, with features defined by edge factors $\phi(Y_i, Y_j)$ and $\phi(Y_i|X_i)$. We used three structures: chains, stars, and grids (Fig. 2.1). Our models had $|Y| = |X|$, and both $P(X)$ and $P(Y|X)$ shared the same structure. Our models used two factor types: associative (in which $\theta_t^* \phi_t(a, b) = s$ if $a = b$ and 0 otherwise) and random (in which each value $\theta_t^* \phi_t(\cdot, \cdot)$ was chosen uniformly at random from the range $[-s, s]$). We call s the factor strength, and we write associative(s) and random(s) for shorthand. Note the strength s is in log-space.

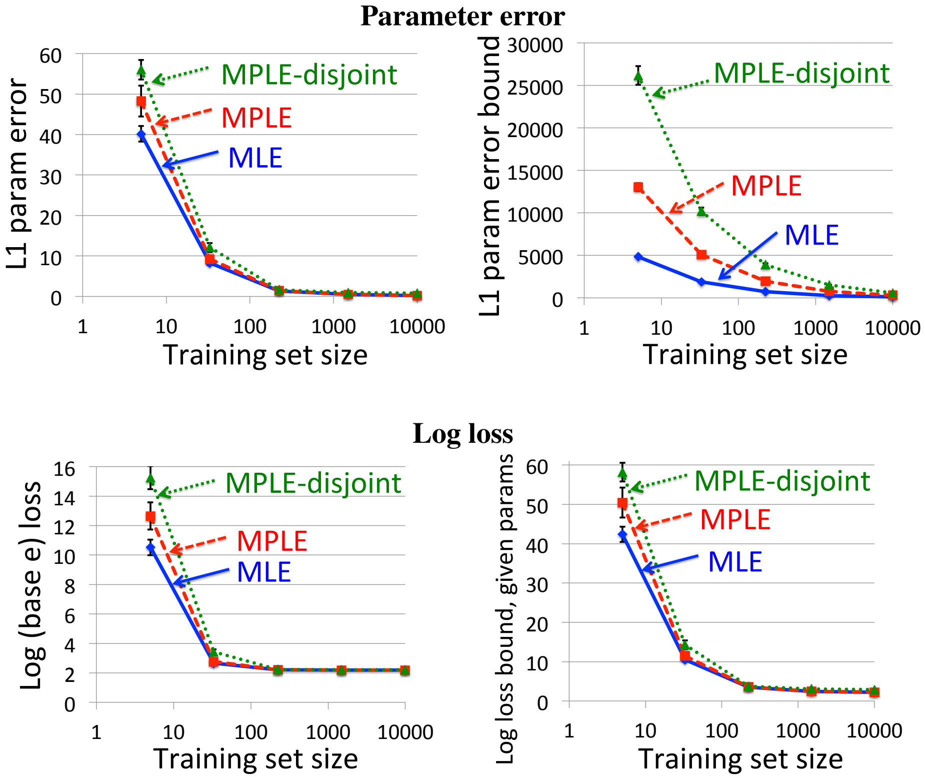


Figure 2.2: **Bounds: training set size.** *Top*: Actual (learned) parameter error (top-left) is much smaller than the bound on parameter error (top-right), but both decrease at similar rates w.r.t. training set size. *Bottom*: The actual log loss (bottom-left) is close to the loss bound given the actual parameter error (bottom-right). (Note log scales on y-axes.) Chains; $|Y| = 4$; random(1) factors. Averaged over 10 models × 10 datasets; error bars 1 stddev.

For optimization, we used conjugate gradient with exact inference for small models in this section and stochastic gradient with approximate inference (Gibbs sampling) for the large models in Sec. 2.5. We chose regularization λ according to each method's sample complexity bound, with $\xi = .5$ (though this choice is technically not valid for small training set sizes).[3] Our results are averaged over 10 runs on different random samples, and 10 models when using random factors.

2.4.2 Comparing Bounds

Our theoretical analysis included two types of bounds: a bound on the parameter estimation error $\|\hat{\theta} - \theta^*\|_1$ in terms of the training set size (Corollary 2.3.2, Corollary 2.3.4, and Theorem 2.3.7 for MLE, MPLE, and MPLE-disjoint, respectively) and a bound on the log loss $\ell_L(\hat{\theta})$ in terms of $\|\hat{\theta} - \theta^*\|_1$ (Theorem 2.3.5). Fig. 2.2 shows that the parameter error bound is much looser than the loss bound for MLE and MPLE with both joint and disjoint optimization. However, both bounds capture the convergence behavior w.r.t. training set size. As expected from our analysis, MPLE performs worse with disjoint optimization than with joint.

[3] We also ran experiments with regularization chosen via k-fold cross validation, which improved results but did not significantly change qualitative comparisons. We omit these results since they do not apply to our analysis.

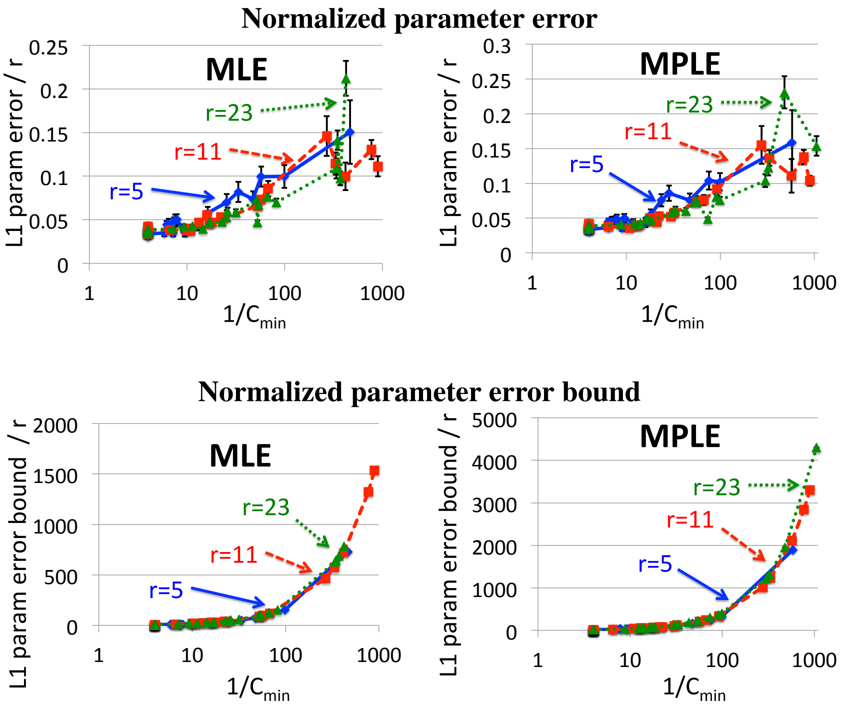


Figure 2.3: **Bounds: minimum eigenvalue C_{min}.** *Top*: Learned parameter error for MLE and MPLE (normalized by dimension r) increases with $1/C_{min}$. *Bottom*: Bound on parameter error for MLE and MPLE increases at a similar rate with $1/C_{min}$. Chains; $|Y| = 2, 4, 8$ ($r = 5, 11, 23$); random(1) factors; 1495 training samples. Each point corresponds to 1 model, averaged 10 datasets; error bars 1 stddev.

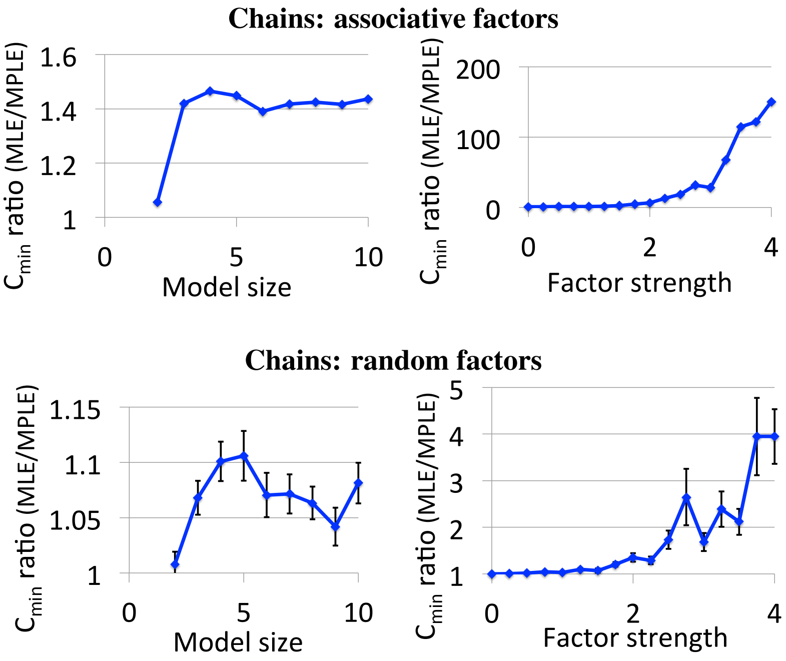
Our bounds for MLE and MPLE depend on a key property: C_{min}. Though C_{min} only needs to lower-bound the Hessian eigenvalues for our analysis, we simplify our discussion from here on by equating C_{min} with the minimum eigenvalue. Our bounds indicate that, for a fixed training set size, the parameter estimation error should be proportional to $1/C_{min}$. Fig. 2.3 plots the error for MLE and MPLE vs. their respective values $1/C_{min}$. Though the bound constants are loose, the $1/C_{min}$ dependence appears accurate.

Our bounds indicate that, for a fixed training set size, the normalized parameter error (normalized by the dimensionality r) should increase only logarithmically in r. Fig. 2.3 plots results for three values of r; increasing r does not significantly effect the normalized error.

This section's results were for chains with random factors, but other model types showed similar behavior. We next study a much wider variety of models.

2.4.3 Eigenspectra

As shown in the previous subsection, C_{min}, the minimum eigenvalue of the loss' Hessian, largely determines learning performance. When choosing a loss, we must trade off the goals of maximizing C_{min} (i.e., maximizing statistical efficiency) and of limiting computational complexity. In this section, we compare the C_{min} for MLE with the C_{min} for MPLE on a range of models, thus offering the reader guidance for when MPLE may replace MLE without sacrificing too much statistical efficiency.


Chains: associative factors

Chains: random factors

Figure 2.4: **Chains: Min eigval ratio MLE/MPLE.** Higher ratios imply MLE is superior. *Left*: The ratio is about constant w.r.t. model size (for fixed factor strength 1). *Right*: The ratio increases with factor strength (for fixed model size $|Y| = 8$). Note: Non-monotonicity in right-most parts of plots is from systematic numerical inaccuracy. Error bars 1 stddev, computed from 100 random models.

Testing model diameter (chains): Fig. 2.4 plots the MLE/MPLE ratio of C_{min} values for chains with associative and random factors. Higher ratios imply lower statistical efficiency for MPLE, relative to MLE. For both, the ratio remains fairly constant as model size increases; i.e., model size does not significantly affect MPLE's relative statistical efficiency. However, increased factor strength decreases MPLE's efficiency, particularly for associative factors.

Testing node degree (stars): Fig. 2.5 compares MLE and MPLE for stars. For both associative and random factors, the C_{min} ratio increases as $|Y|$ increases, indicating that high-degree nodes decrease MPLE's relative statistical efficiency. As with chains, increased factor strength decreases MPLE's efficiency.

Testing treewidth (grids): Fig. 2.6 compares MLE and MPLE for square grids (as in Fig. 2.1). For both factor types, the C_{min} ratio increases as $|Y|$ and factor strength increase. We estimated Hessians for grids with width > 3 by sampling from $P(X)$.

Overall, MPLE appears most statistically efficient for low-degree models with weak variable interactions. In the next section, we discuss how to overcome difficulties with high degree nodes and strong factors by using structured MCLE instead of MPLE.

2.5 Structured Composite Likelihood

MPLE sacrifices statistical efficiency for computational tractability. In this section, we show how to use MCLE to improve upon MPLE's statistical efficiency without much increase in computation. In particular,

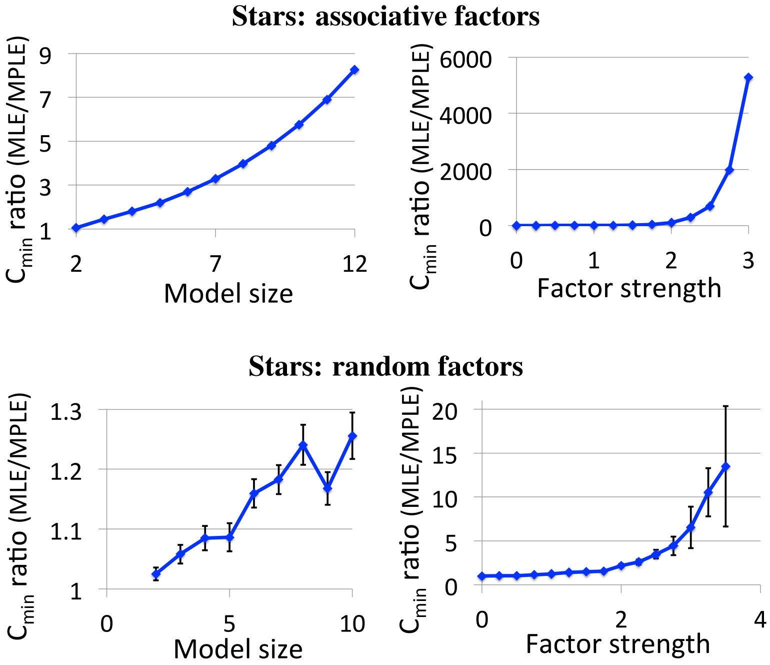


Figure 2.5: **Stars: Min eigval ratio MLE/MPLE.** Higher ratios imply MLE is superior. *Left*: The ratio increases with model size (for fixed factor strength 1). *Right*: The ratio increases with factor strength (for fixed model size $|Y| = 8$). Error bars 1 stddev, computed from 100 random models.

we demonstrate the benefits of careful selection of structured likelihood components for MCLE.

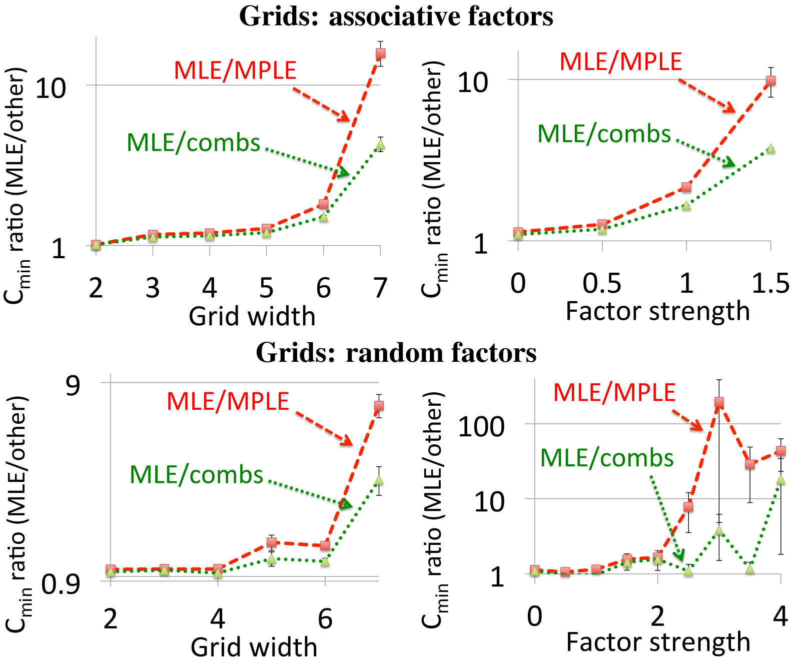
We state two propositions providing a simple method for choosing MCLE components. The first states how to choose a consistent MCLE estimator. The second states that consistent MCLE estimators may be combined to create new, consistent MCLE estimators.

Proposition 2.5.1. *Suppose a set of MCLE components $\mathcal{A}$ covers each Y variable exactly once; i.e., $\cup_i A_i = Y$, and $\sum_i |A_i| = |Y|$. Then the MCLE estimator defined by $\mathcal{A}$ is consistent.*

Proposition 2.5.2. *Suppose two MCLE estimators $\mathcal{A}'$ and $\mathcal{A}''$ are both consistent. Then their union $\mathcal{A} = \mathcal{A}' \cup \mathcal{A}''$ is also a consistent MCLE estimator.*

The simplest MCLE estimator which may be built using Prop. 2.5.1 is MPLE. We advocate the use of *structured* likelihood components, i.e., components A_i containing multiple variables chosen according to the structure of the model. The components should be chosen to ensure that inference is tractable; we restrict ourselves to tree structures, a simple yet very flexible class. We give a simple example in Fig. 2.1, in which two *comb*-like components cover the entire model while maintaining low treewidth (i.e., tractable inference) within each component. In general, MCLE estimators with larger components are more statistically efficient. Fig. 2.6 demonstrates such behavior empirically, with combs (structured MCLE) having higher C_{min} values than MPLE (unstructured MCLE).

Our bound for MCLE indicates that we should choose MCLE estimators based on their components' minimum eigenvalues, but those eigenvalues are often expensive to compute. Corollary 2.3.4 and Prop. 2.5.2 offer a solution: use a mixture of MCLE estimators. Recall that our bound's dependence on ρ_{min} indicates that a mixture of MCLE estimators $\mathcal{A}' \cup \mathcal{A}''$ should have statistical efficiency somewhere in between


Grids: associative factors

Grids: random factors

Figure 2.6: **Grids: Min eigval ratio MLE/MPLE.** Higher ratios imply MLE is superior. We include combs (MCLE) as described in Sec. 2.5; MCLE is strictly superior to MPLE. All y-axes are log-scale. *Left*: The ratio increases with model size (for fixed factor strength 0.5). *Right*: The ratio increases with factor strength (for fixed model size $|Y| = 16$). Error bars 1 stddev, computed from 10 random models.

that of $\mathcal{A}'$ and $\mathcal{A}''$. We present a toy example in Fig. 2.7. This example uses a grid with stronger vertical factors than horizontal ones. As might be expected, MCLE components which include these strong edges (`Combs-vert`) make better estimators than components which do not (`Combs-horiz`). The combination of the two MCLE estimators (`Combs-both`) lies in between.

Our empirical results provide two rules of thumb for choosing a reasonable estimator when lacking expert knowledge: (A) use a small number of structured MCLE components which both cover Y and have low treewidth, and (B) combine multiple such estimators to average out the effects of “bad” components.

Fig. 2.8 gives empirical results on large grids, comparing MLE, MPLE, and comb-structured MCLE. Since we cannot easily compute eigenvalues for large models, we show results in terms of log loss on held-out test data. MCLE achieves much smaller log loss than MPLE, even though their training times are similar.

2.6 The CRF Canonical Parameterization

Our work on composite likelihood above originated with a study of the *canonical parameterization* proposed by Abbeel et al. [2006] for learning parameters of MRFs. We present our findings about the canonical parameterization in this section, as well as how those findings led us to focus on pseudolikelihood (and ultimately composite likelihood).

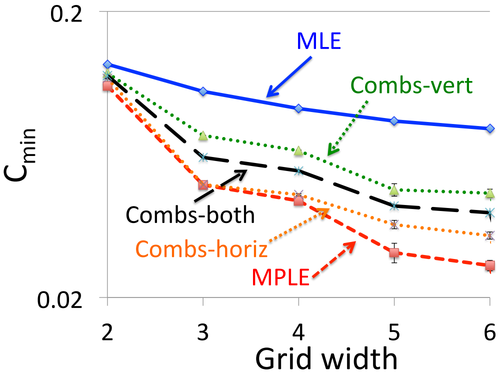


Figure 2.7: **Combining MCLE estimators.** As grid width increases, min eigenvalues C_{min} for estimators decrease (so learning becomes harder). Vertical factors are strongest: associative(1.5) vs. associative$(.5)$. `Combs-vert` is two vertically oriented combs as in Fig. 2.1; `Combs-horiz` is the same combs rotated 90 degrees; and `Combs-both` is their combination. Note C_{min} for `Combs-both` is the average of the C_{min} values for `Combs-vert` and `Combs-horiz`. C_{min} estimated via sampling for width ≥ 4; error bars 1 stddev.

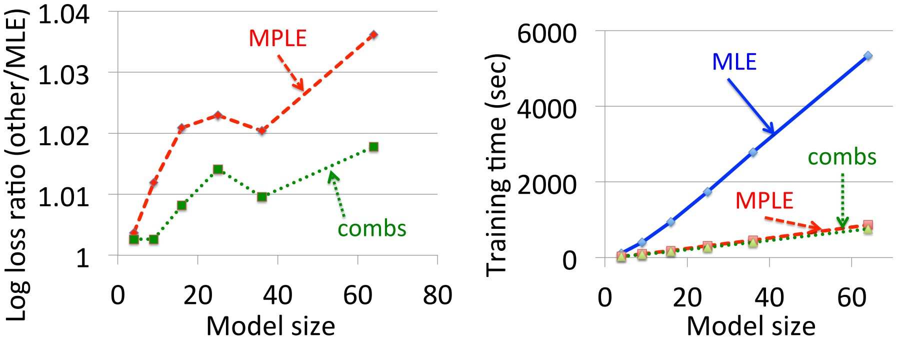


Figure 2.8: **Structured MCLE on grids.** Combs (MCLE) achieves smaller log loss than MPLE *(left)* but uses no more training time *(right)*. Associative$(.5)$ factors; 10,000 training samples.

The method from Abbeel et al. [2006] (Sec. 2.6.1) was remarkable for giving PAC-style bounds for learning parameters of high-treewidth discrete MRFs. Since their method does not use inference during learning and only uses local computation, its computational tractability does not depend on treewidth. It is also data parallel. We approached their work to see if it could be extended to more general classes of models, particularly CRFs.

We first extended their method to CRFs and showed that the polynomial time and sample complexity bounds still hold (Sec. 2.6.2). We then developed improvements to make the method practical for learning with arbitrary structures (Sec. 2.6.3) and to make the parameterization more robust (Sec. 2.6.4).

Finally, we realized that the canonical parameterization essentially reduces to pseudolikelihood (Sec. 2.6.5). Under certain algorithmic design choices, pseudolikelihood learning implicitly computes the canonical parameterization and is much more efficient than explicit computation. We discuss empirical tests on synthetic data backing this claim in Sec. 2.6.6. However, we postulate that, even though the parameterization should not be used algorithmically, it may still be useful as an analytical tool (Sec. 2.6.7).

2.6.1 Background: Canonical Parameterization for MRFs

Contribution: In addition to giving background, we combine results from Abbeel et al. [2006] and Roy et al. [2009] to achieve better computational and sample complexity bounds than in Abbeel et al. [2006].

We first describe the canonical parameterization for MRFs $P(\mathcal{X})$ given by Abbeel et al. [2006]. It is essentially a way of rewriting $P(\mathcal{X})$ in terms of a product of factors, each of which is a conditional probability over a small set of variables. Learning reduces to estimating each factor separately by learning a small conditional distribution from data.

Their method applies to distributions which are representable as bounded-degree factor graphs. A *factor graph* for an MRF is a bipartite graphical representation of a model

$$P(\mathcal{X}) = \tfrac{1}{Z_P} \prod_{j=1}^{J} \phi_j(X_{C_j}) \,, \tag{2.20}$$

with one *variable vertex* per random variable in $\mathcal{X}$ and one *factor vertex* per factor ϕ_j. The factor graph contains an edge between X_i's vertex and ϕ_j's vertex whenever variable X_i is an argument of factor ϕ_j (i.e., $X_i \in X_{C_j}$). Abbeel et al. [2006] show that their method has polynomial time and sample complexity, given that each variable and factor vertex has bounded degree. In other words, each variable must participate in only a small number of factors, and each factor must have a small number of arguments.

Suppose we are given the structure of an MRF, as specified by the factor domains X_{C_j}, as in Eq. (2.20). We call $\{C_j\}_{j=1}^{J}$ the *original factor* domains. The following theorem states the canonical parameterization for MRFs, which tells us how to learn the MRF parameters.

Theorem 2.6.1. ***(MRF Canonical Parameterization)*** *(Theorem 3 and Proposition 4 from Abbeel et al. [2006])*
Assume a distribution $P(\mathcal{X})$ factorizes according to Eq. (2.20), with factor domains indexed by $\{C_j\}_{j=1}^{J}$.

- *Define the* canonical factor domains *as*

$$\{C_j^*\}_{j=1}^{J^*} = \cup_{j=1}^{J} 2^{C_j} \setminus \emptyset, \tag{2.21}$$

  where 2^C is the power set of C.

- *Let $\overline{x}$ be an arbitrary assignment of values to $\mathcal{X}$, called the* reference assignment.

- *Define the j^{th}* canonical factor *as*

$$f_j^*(X_{C_j^*}) = \exp\left(\sum_{U \subseteq C_j^*} (-1)^{|C_j^* \setminus U|} \log P(X_U, \overline{x}_{C_j^* \setminus U} \,|\, \overline{x}_{N_{C_j^*}}) \right) . \tag{2.22}$$

  (This is the "Markov blanket canonical factor" in Section 3.2 of Abbeel et al. [2006].)

Then the distribution $P(\mathcal{X})$ may be equivalently written in the canonical parameterization *as:*

$$P(\mathcal{X}) = P(\overline{x}) \prod_{j=1}^{J^*} f_j^*(X_{C_j^*}) \,. \tag{2.23}$$

Algorithm 2.1: Canonical Parameterization for MRFs [Roy et al., 2009]

```
Input: MRF factor domains $\{C_j\}_{j=1}^{J}$, as in Eq. (2.20); reference assignment $\overline{x}$.
foreach $C_j^* \in \{C_j^*\}_{j=1}^{J^*} = \cup_{j=1}^{J} 2^{C_j} \setminus \emptyset$ do
    Pick any $i_j \in C_j^*$.
    foreach $U \subseteq C_j^*$ do
        Estimate $P(P(X_{i_j}[X_U, \overline{x}_{-U}] \,|\, X_{N_{i_j}}[X_U, \overline{x}_{-U}]))$ from data.
    Compute canonical factor $f_j^*(X_{C_j^*})$, as in Eq. (2.24).
Multiply all canonical factors together to estimate $P(\mathcal{X})$, as in Eq. (2.23).
```

Learning using Theorem 2.6.1 consists of estimating each canonical factor via a set of regression problems $P(X_U, \overline{x}_{C_j^* \setminus U} \,|\, \overline{x}_{N_{C_j^*}})$ and then multiplying the canonical factors together. Abbeel et al. (Theorems 5, 6) show that this parameterization permits parameter learning in polynomial time and sample complexity (polynomial in the size of the original factor graph representation). They also show that slight structure misspecifications do not seriously harm the estimated model (Theorem 7). Since no inference is required, even high-treewidth structures may be learned using polynomial time and samples. Moreover, learning is data parallel since it decomposes into a set of disjoint regression problems over subsets of the variables.

Roy et al. [2009] propose an improvement to this parameterization which reduces the complexity of each canonical factor, paraphrased in the following theorem. First, we define notation: If A is a variable or set of variables and u, v are assignments to disjoint sets of variables U, V s.t. $A \subseteq U \cup V$, then we write $A[u, v]$ to denote an assignment to variables in A taking values from u, v.

Theorem 2.6.2. ***(Improved MRF Canonical Parameterization)*** *(Theorem 2.1 from Roy et al. [2009]) Under the assumptions and definitions from Theorem 2.6.1, additionally define $X_{i_j} \in X_{C_j^*}$ to be an arbitrary variable in the domain of the j^{th} canonical factor. We can equivalently express each canonical factor $f_j^*(X_{C_j^*})$ as:*

$$f_j^*(X_{C_j^*}) = \exp\left(\sum_{U \subseteq C_j^*} (-1)^{|C_j^* \setminus U|} \log P(X_{i_j}[X_U, \overline{x}_{-U}] \,|\, X_{N_{i_j}}[X_U, \overline{x}_{-U}]) \right) . \qquad (2.24)$$

This improved factor involves fewer random variables and so requires less computation and fewer samples to compute than Eq. (2.22). The resulting algorithm is detailed in Alg. 2.1.

Abbeel et al. [2006] use discrete variables and the less efficient parameterization from Eq. (2.22). Theorem 2.6.3 and Theorem 2.6.4, which state computational and sample complexity bounds for Alg. 2.1, generalize Theorems 5 and 6 from Abbeel et al. [2006] by leaving the variable and factor types unspecified. Corollary 2.6.6 and Corollary 2.6.7 instantiate these theorems for models with discrete variables, giving results analogous to Theorems 5 and 6 from Abbeel et al. [2006]. Since Alg. 2.1 uses the more efficient parameterization from Roy et al. [2009], our bounds are better than those from Abbeel et al. [2006]. The bounds measure error using the *Kullback-Liebler (KL) divergence*, which measures distance between two distributions $P(X), Q(X)$:

$$KL(P \| Q) = \mathbf{E}_{P(X)} \left[\log \frac{P(X)}{Q(X)} \right] . \qquad (2.25)$$

The *symmetric KL divergence* is $KL(P \| Q) + KL(Q \| P)$.

Theorem 2.6.3. ***(Computational complexity of the canonical parameterization for MRFs)*** *(Generalized from Abbeel et al. [2006], Roy et al. [2009])*
Suppose that (a) Alg. 2.1 is given the factor graph structure according to which the target distribution $P(\mathcal{X})$ factorizes (Eq. (2.20)); (b) each of the J factors has at most k arguments; and (c) estimating $P(X_{i_j}[X_U, \overline{x}_{-U}] \,|\, X_{N_{i_j}}[X_U, \overline{x}_{-U}])$ for any i_j, U takes time at most C_0. Then Alg. 2.1 runs in time

$$O\left(2^{2k} J \cdot C_0\right) . \tag{2.26}$$

Proof (Theorem 2.6.3):: There are at most $J \cdot 2^k$ canonical factors, each of which has at most 2^k terms $P(X_{i_j}[X_U, \overline{x}_{-U}] \,|\, X_{N_{i_j}}[X_U, \overline{x}_{-U}])$ in Eq. (2.24). ■

Theorem 2.6.4. ***(Sample complexity of the canonical parameterization for MRFs)*** *(Generalized from Abbeel et al. [2006], Roy et al. [2009])*
Abbreviate $P[X_{i_j}; U] \doteq P(X_{i_j}[X_U, \overline{x}_{-U}] \,|\, X_{N_{i_j}}[X_U, \overline{x}_{-U}])$. Let Q be the distribution returned by Alg. 2.1. Suppose that for a given $\epsilon_0, \delta_0 > 0$ and a given $\overline{x}, U, i_j$, we can estimate $P[X_{i_j}; U]$ uniformly well with probability at least $1 - \delta_0$:

$$|\log P[X_{i_j}; U] - \log Q[X_{i_j}; U]| \leq \epsilon_0, \quad \forall X_U . \tag{2.27}$$

Under the assumptions in Theorem 2.6.3, $KL(P\|Q) + KL(Q\|P) \leq 2^{2k+1} J \epsilon_0$ holds with probability at least $1 - 2^{2k} J \delta_0$.

Equivalently, to ensure $KL(P\|Q) + KL(Q\|P) \leq \epsilon$ holds with probability at least $1 - \delta$, we must estimate each $P[X_{i_j}; U]$ uniformly well with $\epsilon_0 = \epsilon / \left(2^{2k+1} J\right)$ and $\delta_0 = \delta / \left(2^{2k} J\right)$.

We first state a lemma which shows how normalization constants cancel when we measure error using symmetric KL divergence.

Lemma 2.6.5. *Suppose two distributions $P(\mathcal{X})$ (as in Eq. (2.20)) and $Q(\mathcal{X}) = \frac{1}{Z_Q} \prod_{j=1}^J \widehat{\phi}_j(X_{C_j})$ share the same set of factor domains $\{C_j\}_{j=1}^J$. Then the symmetric KL divergence factors as:*

$$KL(P\|Q) + KL(Q\|P) \;=\; \sum_j \mathbf{E}_{P(X)}\left[\log \phi_j - \log \widehat{\phi}_j\right] + \mathbf{E}_{Q(X)}\left[\log \widehat{\phi}_j - \log \phi_j\right] . \tag{2.28}$$

Proof (Lemma 2.6.5):: Abbreviate $\phi_j = \phi_j(X_{C_j})$ and $\widehat{\phi}_j = \widehat{\phi}_j(X_{C_j})$.

$$\begin{aligned}
& KL(P\|Q) + KL(Q\|P) && (2.29)\\
& = \mathbf{E}_{P(X)}\left[\log P(X) - \log Q(X)\right] + \mathbf{E}_{Q(X)}\left[\log Q(X) - \log P(X)\right] && (2.30)\\
& = \mathbf{E}_{P(X)}\left[-\log Z_P + \left(\sum_j \log \phi_j\right) + \log Z_Q - \left(\sum_j \log \widehat{\phi}_j\right)\right] \\
& \quad + \mathbf{E}_{Q(X)}\left[-\log Z_Q + \left(\sum_j \log \widehat{\phi}_j\right) + \log Z_P - \left(\sum_j \log \phi_j\right)\right] && (2.31)\\
& = \mathbf{E}_{P(X)}\left[\left(\sum_j \log \phi_j\right) - \left(\sum_j \log \widehat{\phi}_j\right)\right] + \mathbf{E}_{Q(X)}\left[\left(\sum_j \log \widehat{\phi}_j\right) - \left(\sum_j \log \phi_j\right)\right] && (2.32)\\
& = \sum_j \mathbf{E}_{P(X)}\left[\log \phi_j - \log \widehat{\phi}_j\right] + \mathbf{E}_{Q(X)}\left[\log \widehat{\phi}_j - \log \phi_j\right] && (2.33)
\end{aligned}$$

■

Proof (Theorem 2.6.4):: Starting from the result of Lemma 2.6.5, we bound the symmetric KL divergence between the target distribution $P(\mathcal{X})$ and our estimate $Q(\mathcal{X})$. We use the canonical parameterization for both distributions, as defined in Eq. (2.23) and Eq. (2.24). Let $\widehat{f_j}$ denote the j^{th} canonical factor for $Q(\mathcal{X})$.

$$KL(P\|Q) + KL(Q\|P) \;=\; \sum_{j=1}^{J^*} \mathbf{E}_{P(X)}\left[\log f_j - \log \widehat{f_j}\right] + \mathbf{E}_{Q(X)}\left[\log \widehat{f_j} - \log f_j\right] . \tag{2.34}$$

Each expectation may be expanded using the definition of canonical factors in Eq. (2.24). Also using our assumption that conditional probabilities are estimated uniformly well, as in Eq. (2.27), we get

$$\begin{aligned}
\mathbf{E}_{P(X)}\left[\log f_j - \log \widehat{f_j}\right] &= \mathbf{E}_{P(X)}\left[\sum_{U\subseteq C_j^*} (-1)^{|C_j^*\setminus U|}\left(\log P[X_{i_j};U] - \log Q[X_{i_j};U]\right)\right] && (2.35)\\
&\leq \mathbf{E}_{P(X)}\left[\sum_{U\subseteq C_j^*} \left|\log P[X_{i_j};U] - \log Q[X_{i_j};U]\right|\right] && (2.36)\\
&\leq \mathbf{E}_{P(X)}\left[2^{|C_j^*|}\epsilon_0\right] && (2.37)\\
&= 2^{|C_j^*|}\epsilon_0 && (2.38)\\
&\leq 2^k\epsilon_0 . && (2.39)
\end{aligned}$$

(Recall that original and canonical factors have at most k arguments.) Likewise,

$$\mathbf{E}_{Q(X)}\left[\log \widehat{f_j} - \log f_j\right] \leq 2^k\epsilon_0 . \tag{2.40}$$

Using Eq. (2.39), Eq. (2.40) to upper-bound each term in Eq. (2.34), we get the theorem's KL bound:

$$\begin{aligned}
KL(P\|Q) + KL(Q\|P) &= \sum_{j=1}^{J^*} 2^k\epsilon_0 + 2^k\epsilon_0 && (2.41)\\
&= \left(J\cdot 2^k\right)\cdot\left(2^k\epsilon_0 + 2^k\epsilon_0\right) && (2.42)\\
&= J\cdot 2^{2k+1}\epsilon_0 . && (2.43)
\end{aligned}$$

The bound on the probability of failure comes from a union bound over all $2^{2k}J$ regression problems $Q[X_{i_j};U]$. ■

Abbeel et al. [2006] restrict their analysis to the case of discrete variables, for which it is easy to achieve uniform error bounds as in Eq. (2.27). We state results analogous to theirs in the next two corollaries.

Corollary 2.6.6. ***(Computational complexity of the canonical parameterization for discrete MRFs)*** *(Generalized from Abbeel et al. [2006], Roy et al. [2009]) In addition to the assumptions in Theorem 2.6.3, assume that (a) $\mathcal{X}$ are discrete variables with maximum arity v and (b) Alg. 2.1 is given n i.i.d. training examples generated from $P(\mathcal{X})$. Let $b \doteq 1 + \max_i |N_i|$ be the maximum number of neighbors of any variable, plus 1. Then Alg. 2.1 runs in time*

$$O\left(2^{2k}J\left(v^k + nb\right)\right) . \tag{2.44}$$

Proof (Corollary 2.6.6):: Each local distribution $Q[X_{i_j}; U] = Q(X_{i_j}[X_U, \overline{x}_{-U}] \,|\, X_{N_{i_j}}[X_U, \overline{x}_{-U}])$ may be represented as a table of counts over k variables. (Recall that $|U| \leq k$ variables are undetermined, while the rest are set as $\overline{x}$.) This table has at most v^k values, so initializing it to zero takes time $O(v^k)$. Adding counts to this table for all training examples takes time $O(nb)$ since each local distribution depends on the values of at most b variables. Normalizing the table takes time $O(v^k)$. So $C_0 = v^k + nb$ in Theorem 2.6.3. ■

Theorem 5 in Abbeel et al. [2006] gives a running time bound of $O(n2^k J(k+b) + 2^{2k} Jv^k)$; their proof does not account for initializing and normalizing each $Q[X_{i_j}; U]$ in the canonical factors. However, this difference is relatively unimportant since the $2^{2k} Jv^k$ term dominates in many settings. The following corollary, bounding the sample complexity, gives a result essentially the same as that in Abbeel et al. [2006].

Corollary 2.6.7. ***(Sample complexity of the canonical parameterization for discrete MRFs)*** *(Generalized from Abbeel et al. [2006], Roy et al. [2009]) In addition to the assumptions in Theorem 2.6.3 and Corollary 2.6.6, assume $\gamma \doteq \min_x P(x_i, x_{N_i}) > 0$. To ensure $KL(P\|Q) + KL(Q\|P) \leq \epsilon$ holds with probability at least $1 - \delta$, it suffices to use n i.i.d. examples from $P(\mathcal{X})$, where*

$$\begin{aligned} n &\geq \frac{2^{4k+3} J^2 \big(1+\epsilon/\big(2^{2k+2}J\big)\big)^2}{\gamma^2 \epsilon^2} \log \frac{2^{2k+2} J v^b}{\delta} && (2.45) \\ &\approx \frac{2^{4k+3} J^2}{\gamma^2 \epsilon^2} \log \frac{2^{2k+2} J v^b}{\delta} \,. && (2.46) \end{aligned}$$

Proof (Corollary 2.6.7):: Recall that variables have arity at most v, and $b = \max_i |N_i| + 1$. Lemma 18 in Abbeel et al. [2006] tells us that, for $|\log P(x_i|x_{N_i}) - \log Q(x_i|x_{N_i})| \leq \epsilon_0$ to hold with probability at least $1 - \delta_0$ for all values x_i, x_{N_i}, we need at most $n \geq \frac{(1+\epsilon_0/2)^2}{2\gamma^2(\epsilon_0/2)^2} \log \frac{4v^b}{\delta_0}$ i.i.d. examples.

Theorem 2.6.4 tells us how to substitute ϵ_0, δ_0 for values in terms of ϵ, δ. Plugging these values in, we get:

$$\begin{aligned} n &\geq \frac{\big(1+\epsilon/\big(2^{2k+2}J\big)\big)^2}{2\gamma^2 \big(\epsilon/\big(2^{2k+2}J\big)\big)^2} \log \frac{4v^b}{\delta/\big(2^{2k}J\big)} && (2.47) \\ &= \frac{2^{4k+3} J^2 \big(1+\epsilon/\big(2^{2k+2}J\big)\big)^2}{\gamma^2 \epsilon^2} \log \frac{2^{2k+2} J v^b}{\delta}. && (2.48) \end{aligned}$$

■

The main distinctions between our bounds for discrete models and those from Abbeel et al. [2006] are the definitions of b and γ. For us, b bounds the size of the Markov blanket (neighbors) of each variable: $b = \max_i |N_i| + 1$. For Abbeel et al. [2006], b bounds the size of the Markov blanket of sets of variables: $b = \max_j |N_{C_j}|$, where $\{C_j\}$ are the domains of the original factors. Since C_j can be of size k, their b can be about k times larger than ours. More significantly, their γ (γ^{k+b} in their notation) is much smaller; in their notation, our bound would have γ^{k+1} instead of γ^{k+b}, giving a much smaller sample complexity. Both of these improvements come from the simplified canonical factor in Eq. (2.24) from Roy et al. [2009].

Roy et al. [2009] demonstrate empirically that their simplification permits the canonical parameterization to be used on real-world problems. Nevertheless, as the degree of the factor graph increases, this learning method's computational and sample complexity can increase at an exponential rate, thus prohibiting its use on many models. In fact, even for simple, low-treewidth models for which traditional max-likelihood parameter learning is efficient, a single variable involved in many factors prevents the use of this canonical parameterization. In Sec. 2.6.3, we show how to modify this method to make it practical for arbitrary structures.

Together, Corollary 2.6.6 and Corollary 2.6.7 prove PAC learnability for positive distributions representable as bounded-degree factor graphs. At the end of Sec. 2.6.5, we compare this learnability result with our results using pseudolikelihood and composite likelihood from Sec. 2.3.

2.6.2 Extension to CRFs

Contribution: We extend the canonical parameterization to CRFs, generalizing the sample and computational complexity results from Sec. 2.6.2.

Suppose the target distribution is representable as a bounded-degree factor graph:

$$P(\mathcal{Y}|\mathcal{X}) = \tfrac{1}{Z_P(\mathcal{X})} \prod_{j=1}^{J} \phi_j(Y_{C_j}, X_{D_j}) \,. \tag{2.49}$$

For CRFs, the factor graph only has variable vertices for $\mathcal{Y}$, not $\mathcal{X}$. Therefore, the complexity of learning will largely depend on $\mathcal{Y}$, not on $\mathcal{X}$.

The extension of the canonical parameterization from MRFs to CRFs is straightforward. Essentially, all probabilities in the MRF parameterization and proofs may be modified to condition on all of $\mathcal{X}$. At the end, the probabilities may be simplified by conditioning on the Markov blanket rather than all of $\mathcal{X}$: $P(A|B,\mathcal{X}) = P(A|B, X_{N_A})$.

The following theorem generalizes the canonical parameterization to CRFs. We overload the notation N_C to index neighbors of Y_C in both $\mathcal{Y}$ and $\mathcal{X}$; i.e., Y_{N_C} is the Markov blanket of Y_C in $\mathcal{Y}$, and X_{N_C} is the Markov blanket of Y_C in $\mathcal{X}$.

Theorem 2.6.8. ***(CRF Canonical Parameterization)***
Assume a distribution $P(\mathcal{Y}|\mathcal{X})$ factorizes according to Eq. (2.49), with factor domains indexed by $\{C_j\}_{j=1}^{J}$.

- *Define the canonical factor domains $\{C_j^*\}_{j=1}^{J^*}$ as in Eq. (2.21).*
- *Let the reference assignment $\overline{y}$ be an arbitrary assignment of values to $\mathcal{Y}$ (not to $\mathcal{X}$).*
- *Define the j^{th}* canonical factor *to be*

$$f_j^*(Y_{C_j^*}, X_{D_j^*}) = \exp\left(\sum_{U \subseteq C_j^*} (-1)^{|C_j^* \setminus U|} \log P\left(Y_{i_j}[Y_U, \overline{y}_{-U}] \,\middle|\, Y_{N_{i_j}}[Y_U, \overline{y}_{-U}], X_{N_{i_j}} \right) \right) , \tag{2.50}$$

where $i_j \in C_j^*$ may be chosen arbitrarily. (Note D_j^* is defined w.r.t. i_j: $X_{D_j^*} = X_{N_{i_j}}$.)

Then the distribution $P(\mathcal{Y}|\mathcal{X})$ may be equivalently written in the CRF canonical parameterization *as:*

$$P(\mathcal{Y}|\mathcal{X}) = P(\overline{y}|X) \prod_{j=1}^{J^*} f_j^*(Y_{C_j^*}, X_{D_j^*}) \,. \tag{2.51}$$

Algorithm 2.2: Canonical Parameterization for CRFs

```
Input: CRF factor domains $\{(C_j, D_j)\}_{j=1}^J$, as in Eq. (2.49); reference assignment $\overline{y}$.
foreach $C_j^* \in \{C_j^*\}_{j=1}^{J^*} = \cup_{j=1}^J 2^{C_j} \setminus \emptyset$ do
    Pick any $i_j \in C_j^*$.
    foreach $U \subseteq C_j^*$ do
        Estimate $P\left(Y_{i_j}[Y_U, \overline{y}_{-U}] \,\middle|\, Y_{N_{i_j}}[Y_U, \overline{y}_{-U}], X_{N_{i_j}}\right)$ from data.
    Compute canonical factor $f_j^*(Y_{C_j^*}, X_{D_j^*})$, as in Eq. (2.50).
Multiply all canonical factors together to estimate $P(\mathcal{Y}|\mathcal{X})$, as in Eq. (2.51).
```

Proof Sketch (Theorem 2.6.8): The proof is exactly the same as for MRFs in Abbeel et al. [2006] and Roy et al. [2009], except that all probabilities over $\mathcal{Y}$ are modified to be conditioned on $\mathcal{X}$: $P(Y_A|Y_B) \longrightarrow P(Y_A|Y_B, \mathcal{X})$. At the end, simplify the conditional probabilities in the theorems by conditioning on the Markov blankets instead of all of $\mathcal{X}$: $P(Y_A|Y_B, \mathcal{X}) \longrightarrow P(Y_A|Y_B, X_{N_A})$. ■

As for MRFs, learning CRF parameters requires estimating the conditional distributions in the canonical factors. Importantly, the domains of canonical factors are defined w.r.t. the original factors' domains in $\mathcal{Y}$, not in $\mathcal{X}$. The algorithm is detailed in Alg. 2.2.

The computational and sample complexity bounds for MRFs generalize to CRFs. Given a bounded-degree factor graph CRF structure, we can learn its parameters in polynomial time and sample complexity. The following two theorems generalize Theorem 2.6.3 and Theorem 2.6.4 from MRFs to CRFs.

Theorem 2.6.9. ***(Computational complexity of the canonical parameterization for CRFs)*** *Abbreviate $P[Y_{i_j}; U] \doteq P\left(Y_{i_j}[Y_U, \overline{y}_{-U}] \,\middle|\, Y_{N_{i_j}}[Y_U, \overline{y}_{-U}], X_{N_{i_j}}\right)$. Suppose that (a) Alg. 2.2 is given the factor graph structure according to which the target distribution $P(\mathcal{Y}|\mathcal{X})$ factorizes (Eq. (2.49)); (b) each of the J factors has at most k arguments in $\mathcal{Y}$; and (c) estimating $P[Y_{i_j}; U]$ for any i_j, U takes time at most C_0. Then Alg. 2.2 runs in time*

$$O\left(2^{2k} J \cdot C_0\right) . \tag{2.52}$$

Theorem 2.6.10. ***(Sample complexity of the canonical parameterization for CRFs)*** *Let Q be the distribution returned by Alg. 2.2. Suppose that for a given $\epsilon_0, \delta_0 > 0$ and a given $\overline{y}, U, i_j$, we can estimate $P[Y_{i_j}; U]$ uniformly well with probability at least $1 - \delta_0$:*

$$|\log P[Y_{i_j}; U] - \log Q[Y_{i_j}; U]| \leq \epsilon_0, \quad \forall X_U . \tag{2.53}$$

Under the assumptions in Theorem 2.6.9, $KL(P\|Q) + KL(Q\|P) \leq 2^{2k+1} J\epsilon_0$ holds with probability at least $1 - 2^{2k} J\delta_0$.

Equivalently, to ensure $KL(P\|Q) + KL(Q\|P) \leq \epsilon$ holds with probability at least $1 - \delta$, we must estimate each $P[Y_{i_j}; U]$ uniformly well with $\epsilon_0 = \epsilon / \left(2^{2k+1} J\right)$ and $\delta_0 = \delta / \left(2^{2k} J\right)$.

The proofs of Theorem 2.6.9 and Theorem 2.6.10 are exactly the same as the proofs for MRFs in Theorem 2.6.3 and Theorem 2.6.4. We do not add corollaries for discrete models here, but they would be analogous to those for MRFs in Corollary 2.6.6 and Corollary 2.6.7.

The canonical parameterization for general models is unintuitive, so we write out the parameterization for pairwise models in the following corollary to help the reader's understanding.

Algorithm 2.3: Canonical Parameterization for CRFs, with Pre-computed $P(Y_i \mid Y_{N_i}, X_{N_i})$

```
Input: CRF factor domains $\{(C_j, D_j)\}_{j=1}^J$, as in Eq. (2.49); reference assignment $\bar{y}$.
foreach $Y_i \in \mathcal{Y}$ do
    Estimate $P(Y_i \mid Y_{N_i}, X_{N_i})$ from data.
foreach $C_j^* \in \{C_j^*\}_{j=1}^{J^*} = \cup_{j=1}^J 2^{C_j} \setminus \emptyset$ do
    Pick any $i_j \in C_j^*$.
    Compute canonical factor $f_j^*(Y_{C_j^*}, X_{D_j^*})$, as in Eq. (2.50), using pre-computed
    $P(Y_{i_j} \mid Y_{N_{i_j}}, X_{N_{i_j}})$.
Multiply all canonical factors together to estimate $P(\mathcal{Y}|\mathcal{X})$, as in Eq. (2.51).
```

Corollary 2.6.11. ***(Canonical Parameterization for Pairwise CRFs)***
Suppose a distribution $P(\mathcal{Y}|\mathcal{X})$ is expressible as a CRF as in Eq. (2.49), with only pairwise factors: $|C_j| = 2$, $\forall j$. We write $Edges \doteq \{C_j\}_{j=1}^J$. The distribution may be expressed as:

$$\begin{aligned}
&\log P(\mathcal{Y}|\mathcal{X}) \qquad (2.54)\\
&\quad = \log P(\bar{y}|\mathcal{X})\\
&\qquad + \sum_i \log P\left(Y_i \mid \bar{y}_{N_i}, X_{N_i}\right)\\
&\qquad + \sum_{(i,j)\in Edges} \log P(Y_i \mid Y_j, \bar{y}_{N_i \setminus j}, X_{N_i}) - \log P(\bar{y}_i \mid Y_j, \bar{y}_{N_i \setminus j}, X_{N_i}) - \log P(Y_i \mid \bar{y}_{N_i}, X_{N_i}),
\end{aligned}$$

where we pick an endpoint i from each (i, j) arbitrarily.

2.6.3 Extension to Arbitrary Structures

Contribution: We mention two improvements to the canonical parameterization so that it is practical for learning with arbitrary structures.

First, in the improved canonical factors (Eq. (2.50) for CRFs and Eq. (2.24) and MRFs), note that any $i_j \in C_j^*$ is acceptable, so it makes sense to pre-compute the local conditional distributions $P(Y_i \mid Y_{N_i}, X_{N_i})$ for all i. In practice, learning these $|\mathcal{Y}|$ distributions (with no fixed variables) is more efficient than learning $2^{2k}J$ distributions (with variables fixed to $\bar{y}$), for each term in the canonical factors. (Roy et al. [2009], who focus on structure learning, do not explicitly discuss this pre-computation.) A modified algorithm which pre-computes the local conditional distributions is given in Alg. 2.3.

Second, we can *sub-factorize* the conditional probabilities $P(Y_i \mid Y_{N_i}, X_{N_i})$. Since we already assume knowledge of the target distribution's structure, we can factorize these local conditional probabilities according to the structure.

These improvements result in a learning method with these properties:

- The number of parameters which must be estimated for the canonical parameterization is within a constant factor of the number of parameters in the traditional representation.
- The canonical parameterization has the *same structure* (set of factors) as the target distribution.

Unfortunately, it is not straightforward to use these ideas to improve our computational and sample complexity bounds. While the general results in Theorem 2.6.9 and Theorem 2.6.10 still hold, it is difficult to instantiate better bounds for specific classes of models. Our bounds for discrete MRFs in Corollary 2.6.6 and Corollary 2.6.7 require that local conditional distributions be estimated uniformly well (Eq. (2.27)). Achieving these uniform bounds on general models is not easy, even when there are bounds w.r.t. KL divergence or other average-case measures.

2.6.4 Optimizing the Reference Assignment

Contribution: We discuss the choice of the reference assignment, with empirical benefits and implications for theoretical analysis (discussed in Sec. 2.6.7).

The canonical parameterization in Eq. (2.51) is defined with respect to an arbitrary reference assignment $\overline{y}$, and current theory is oblivious to the choice of $\overline{y}$. In practice, the choice of $\overline{y}$ is important. Intuitively, if a value $\overline{y}_j$ is rare in our data, then our probability estimates involving $\overline{y}_j$ are likely to be inaccurate.

To avoid a worst-case scenario, we could choose $\overline{y}$ randomly. A better choice might be per-variable MLE: set each $\overline{y}_j$ as the empirical maximum likelihood estimate.

Rather than fixing $\overline{y}$ to a single value, we can also average over an arbitrary distribution $Q(\overline{Y})$. When we express the canonical parameterization in log form, the parameterization decomposes into a sum of log conditional probabilities. Taking the expectation over $\overline{y}$ w.r.t. $Q(\overline{Y})$ is equivalent to a geometric mean in non-log-space. The following theorem gives the canonical parameterization with reference assignment averaging.

Theorem 2.6.12. ***(CRF Canonical Parameterization, with reference assignment averaging)***

Choose an arbitrary distribution $Q(\overline{Y})$. Under the assumptions in Theorem 2.6.8, the distribution $P(\mathcal{Y}|\mathcal{X})$ may be expressed by averaging over reference assignments $\overline{y}$ (in log space):

$$\begin{aligned}
&\log P(\mathcal{Y}|\mathcal{X}) \qquad (2.55)\\
&\quad = \mathbf{E}_{Q(\overline{Y})}\left[\log P(\overline{Y}|X)\right]\\
&\qquad + \sum_{j=1}^{J^*} \sum_{U \subseteq C_j^*} (-1)^{|C_j^* \setminus U|} \mathbf{E}_{Q(\overline{Y})}\left[\log P\left(Y_{i_j}[Y_U, \overline{Y}_{-U}] \,\middle|\, Y_{N_{i_j}}[Y_U, \overline{Y}_{-U}], X_{N_{i_j}}\right)\right] .
\end{aligned}$$

Proof Sketch (Theorem 2.6.12): Take the expectation w.r.t. $Q(\overline{Y})$ of the canonical parameterization in Eq. (2.51), and use linearity of expectation. ■

Joseph B.: I commented out the above theorem instantiated for pairwise models.

Since we can take the expectation over $\overline{Y}$ separately for each conditional probability (in log space), reference assignment averaging can be done efficiently for many factor representations (including table factors for discrete variables and Gaussian factors for real variables). We propose two distributions $Q(\overline{Y})$: (a) the uniform distribution and (b) the training sample distribution.

Abbeel et al. [2006] and Roy et al. [2009] did not discuss how to choose the reference assignment. However, after we derived these methods, Pieter Abbeel told us in a conversation that he had considered similar ideas, though those results remain unpublished. His published work on this parameterization came before the improvements of Roy et al. [2009], which were important for making the method practical.

2.6.5 Relation to Pseudolikelihood

The canonical parameterization with pre-computed local conditional distributions in Alg. 2.3 might remind the reader of pseudolikelihood. This section elucidates the relationship between the two methods. The crucial choice in the canonical parameterization is in how we calculate the local conditional distributions which are used to compute canonical factors.

We first show that, under one choice for calculating these local conditional distributions, the canonical parameterization reduces to a maximum pseudolikelihood estimate. We then demonstrate other choices, under which the canonical parameterization and pseudolikelihood are no longer equivalent. Finally, we argue that the latter choices are non-optimal and that pseudolikelihood will generally outperform the canonical parameterization. Our experiments discussed in Sec. 2.6.6 support this argument.

The following theorem states that, with reasonable choices of how to estimate the local conditional distributions, the canonical parameterization produces the same result as the maximum pseudolikelihood estimate.

Theorem 2.6.13. ***(CRF Canonical Parameterization: Reduction to Pseudolikelihood)***

Under the assumptions in Theorem 2.6.8, additionally assume:

- *we use Alg. 2.3, which pre-computes local conditional distributions $P(Y_i \,|\, Y_{N_i}, X_{N_i})$, and*
- *we compute $P(Y_i \,|\, Y_{N_i}, X_{N_i})$ using either (a) joint optimization with shared parameters or (b) disjoint optimization with factor averaging, as described in Sec. 2.3.4.*

Then the canonical parameterization produces the same parameter estimates as pseudolikelihood.

Proof (Theorem 2.6.13): We assumed that our estimated local conditional distribution $\widehat{P}(Y_i \,|\, Y_{N_i}, X_{N_i})$, $\forall i$, have consistent parameters: if $\widehat{P}(Y_i \,|\, Y_{N_i}, X_{N_i})$ and $\widehat{P}(Y_j \,|\, Y_{N_j}, X_{N_j})$ both depend on a factor $\phi_k(Y_i, Y_j, \cdot)$, then both local conditional distributions use the same parameters (factor values for ϕ_k). Since we have a single estimate of the parameters for each factor ϕ_k, we can plug these into Eq. (2.49) to produce an estimate $\widehat{P}(\mathcal{Y}|\mathcal{X})$ of the full distribution; this is exactly the maximum pseudolikelihood estimate.

We now need to show that using our estimates $\widehat{P}(Y_i \,|\, Y_{N_i}, X_{N_i})$ to compute the canonical factors in Eq. (2.50) and then plugging those canonical factors into Eq. (2.51) produces the same estimate $\widehat{P}(\mathcal{Y}|\mathcal{X})$ of the full distribution. This equivalence is exactly what Theorem 2.6.8 states, where we replace P in Theorem 2.6.8 with our estimate $\widehat{P}$. ■

We presented Theorem 2.6.13 in Bradley and Guestrin [2012]. This theorem is really a restatement of Theorem 2.6.8, but as we argue later, this theorem's assumptions are very practical.

We now give an opposing result, explaining that the two methods are not necessarily equivalent under modified assumptions. We still assume that we estimate local conditional distributions $\widehat{P}(Y_i|Y_{N_i}, X_{N_i})$ for each i, rather separately estimating $P(Y_{i_j}[Y_U, \bar{y}_{-U}] \,\big|\, Y_{N_{i_j}}[Y_U, \bar{y}_{-U}], X_{N_{i_j}})$ for each i_j and U in Eq. (2.50). The canonical parameterization uses these estimates to compute the canonical factors in Eq. (2.50); the pseudolikelihood estimate uses factor averaging, as described in Sec. 2.3.4.

The key to the following theorem is that we permit conflicting factor estimates. If $P(Y_i|Y_{N_i}, X_{N_i})$ depends upon a factor ϕ_k, denote the factor estimate in $\widehat{P}(Y_i|Y_{N_i}, X_{N_i})$ as $\widehat{\phi}_k^{(i)}$. An inconsistent factor estimate is a pair $\widehat{\phi}_k^{(i)}(y_{C_k}) \neq \widehat{\phi}_k^{(j)}(y_{C_k})$, where $i, j \in C_k$ and y_{C_k} are some values for Y_{C_k}.

Theorem 2.6.14. ***(CRF Canonical Parameterization: Distinction from Pseudolikelihood)***

Under the assumptions in Theorem 2.6.8, as well as the assumptions that:

- *the canonical factors in Eq. (2.50) are computed using local conditional distributions $\widehat{P}(Y_i|Y_{N_i}, X_{N_i})$ estimated separately for each i and*
- *there exists at least one factor $\phi_k(Y_{C_k}, X_{D_k})$ s.t. $|Y_{C_k}| > 1$,*

there exist:

- *a set of local conditional distributions with conflicting factor estimates and*
- *a choice of i_j for each canonical factor j*

such that the canonical parameterization and pseudolikelihood estimate produce different estimates of $P(\mathcal{Y}|\mathcal{X})$.

Proof (Theorem 2.6.14): Sec. A.2.1

The proof relies heavily on the fact that the j^{th} canonical factor in Eq. (2.50) is defined w.r.t. an arbitrary $i_j \in C_j^*$. We now show that the canonical parameterization can be modified to average over all $i_j \in C_j^*$ but that this modification does not make it equivalent to pseudolikelihood when there are conflicting factor estimates.

Theorem 2.6.15. ***(CRF Canonical Parameterization, with i_j averaging)***

Under the assumptions in Theorem 2.6.8, the canonical factors may be expressed by averaging over all $i_j \in C_j^*$ (in log space):

$$\begin{aligned}
&\log P(\mathcal{Y}|\mathcal{X}) \qquad (2.56)\\
&\quad = \log P(\overline{y}|X)\\
&\qquad + \sum_{j=1}^{J^*} \tfrac{1}{|C_j^*|} \sum_{i_j \in C_j^*} \sum_{U \subseteq C_j^*} (-1)^{|C_j^* \setminus U|} \log P\left(Y_{i_j}[Y_U, \overline{y}_{-U}] \,\middle|\, Y_{N_{i_j}}[Y_U, \overline{y}_{-U}], X_{N_{i_j}} \right) .
\end{aligned}$$

Proof Sketch (Theorem 2.6.15): Since $i_j \in C_j^*$ may be chosen arbitrarily for each canonical factor j, we may take the expectation over i_j w.r.t. an arbitrary distribution. In this case, we use the uniform distribution over i_j. ■

Theorem 2.6.16. ***(CRF Canonical Parameterization, with i_j averaging: Distinction from Pseudolikelihood)***

Under the assumptions in Theorem 2.6.15, as well as the assumptions that:

- *the canonical factors in Eq. (2.50) are computed using local conditional distributions $\widehat{P}(Y_i|Y_{N_i}, X_{N_i})$ estimated separately for each i and*
- *there exists at least one factor $\phi_k(Y_{C_k}, X_{D_k})$ s.t. $|Y_{C_k}| > 1$,*

there exists a set of local conditional distributions with conflicting factor estimates such that the canonical parameterization and pseudolikelihood estimate produce different estimates of $P(\mathcal{Y}|\mathcal{X})$.

Proof (Theorem 2.6.16): Sec. A.2.2

Since the canonical parameterization can be distinct from pseudolikelihood, the obvious question is: which is better? We list two main reasons why we believe that pseudolikelihood is preferable.

- *Pseudolikelihood estimation involves less computation.* The canonical parameterization may either (a) use pseudolikelihood as a subroutine for calculating the canonical factors or (b) compute each conditional probability in Eq. (2.50) separately, which (based on our experiments) is much more computationally costly.
- *Pseudolikelihood with joint optimization may give better estimates.* The sample complexity bounds for pseudolikelihood with joint and disjoint optimization in Sec. 2.3.4 and Sec. 2.3.2 indicate that joint optimization may give more accurate results than disjoint optimization. (These are only upper bounds, but our experiments discussed in Sec. 2.4 match this prediction.) The canonical parameterization uses disjoint optimization (else it reduces to pseudolikelihood). We have not found evidence (theoretical or empirical) that the combination of disjoint estimates of local conditional probabilities used by the canonical parameterization is superior to the combination used by pseudolikelihood with factor averaging (let alone pseudolikelihood with joint optimization).

PAC Learnability Results

We briefly compare the PAC learnability results based on the canonical parameterization with those based on pseudolikelihood (MPLE) and composite likelihood (MCLE). The class of models currently known to be PAC-learnable using the canonical parameterization is the class of strictly positive distributions representable as bounded-degree factor graphs (Corollary 2.6.6, Corollary 2.6.7). We postulate that this class is a strict subset of the class of models PAC-learnable using MCLE, barring issues of identifiability.

There are clearly models outside of this class which are learnable using MCLE. For example, star-structured models are not representable as bounded-degree factor graphs, but it is a class of models learnable via MCLE (if factor strengths are bounded). (On star graphs, MCLE may be chosen to be equivalent to MLE, for which our sample complexity bound in Theorem 2.3.1 is a PAC bound. The quantity C_{min} increases at most polynomially with model size.)

To prove our postulate, we would need to prove that the class of identifiable positive bounded-degree factor graph models is PAC-learnable using MCLE. While we do not have a thorough proof, we present a rough argument for why this class is learnable using MPLE. The canonical parameterization requires uniformly accurate estimates of conditional probabilities $P(X_i|X_{N_i})$. Since the model is identifiable, accurate estimates of conditional probabilities (which imply low log loss) imply accurate parameter estimates. If regressing each variable on its neighbors produces accurate parameter estimates, then MPLE will succeed.

It is interesting that the canonical parameterization overcomes identifiability issues. It would be valuable if we could prove that MPLE succeeds (w.r.t. log loss) on non-identifiable models.

2.6.6 Experiments

We ran parameter learning experiments comparing pseudolikelihood, MLE, and the canonical parameterization, including all four methods for choosing the reference assignment $\bar{y}$: random, per-variable MLE, uniform expectation, and sample expectation. We also tested the canonical parameterization with and without sub-factorization (Sec. 2.6.3). We tested discrete and real-valued (Gaussian) synthetic data, both of which gave similar results.

We mention our main findings qualitatively but do not give details because of our later discovery in Sec. 2.6.7 that the canonical parameterization essentially reduces to pseudolikelihood. We discuss algorithm performance in terms of convergence to the optimal held-out log likelihood as the sample size n increases, and we compare with MLE, which serves as a gold standard (since the models were low-treewidth, permitting exact inference).

- Sub-factorization (Sec. 2.6.3) improved performance of the canonical parameterization. Even with the extra computation required for sub-factorization, learning was much faster than MLE training.
- Taking expectations over reference assignments $\overline{y}$ (Sec. 2.6.4) improved performance over using fixed $\overline{y}$.
- When using both sub-factorization and reference assignment averaging, the canonical parameterization performed almost as well as MLE, and it performed very similarly to pseudolikelihood (explained in Sec. 2.6.7).

2.6.7 Applications in Analysis

In the previous section, we argued that the canonical parameterization is not a useful algorithm since pseudolikelihood is preferable. In this section, we suggest that the canonical parameterization may still be useful as a tool for theoretical analysis.

We postulate that the canonical parameterization may be useful for analyzing pseudolikelihood, or any parameter estimation method which produces a single set of factor estimates (without the conflicting factors discussed in Sec. 2.6.5). Like Abbeel et al. [2006], we suggest using the canonical parameterization to rewrite the error in estimating the full distribution in terms of the error in estimating local conditional probabilities. However, we use the ideas of averaging over both reference assignments and $i_j \in C_j^*$, permitting us to express the local errors differently. The final step—showing how to bound the local errors in a nontrivial way—still eludes us. Nevertheless, we argue that our local error terms should be easier to bound (requiring fewer training examples) than the uniform error bound used by Abbeel et al. [2006] in Eq. (2.27).

In the following, we use pairwise MRFs for clarity of presentation, but general CRFs may be analyzed analogously. The following theorem bounds the symmetric KL divergence between two distributions using the canonical parameterization with averaged reference assignments and i_j.

Theorem 2.6.17. *Given $P \doteq P(Y)$ and $Q \doteq Q(Y)$ which factorize according to Eq. (2.20), denote the local conditionals of P by $P_i \doteq P(Y_i|Y_{N_i})$; likewise, denote $Q_i \doteq Q(Y_i|Y_{N_i})$. We may write the symmetric KL divergence between P and Q as:*

$$\begin{aligned} & KL(P\|Q) + KL(Q\|P) \qquad (2.57) \\ & = \sum_i \left(1 - \tfrac{|N_i|}{2}\right) \mathbf{E}_{P(Y_{N_i})} \Big[\mathbf{E}_{P(Y_i|Y_{N_i})} \left[\log P_i - \log Q_i\right] + \mathbf{E}_{Q(Y_i)} \left[\log Q_i - \log P_i\right] \Big] \\ & \qquad + \tfrac{1}{2} \sum_{j \in N_i} \mathbf{E}_{Q(Y_j)P(Y_{N_i \setminus j})} \Big[\mathbf{E}_{P(Y_i|Y_{N_i \setminus j})} \left[\log P_i - \log Q_i\right] + \mathbf{E}_{Q(Y_i|Y_j)} \left[\log Q_i - \log P_i\right] \Big] \ . \end{aligned}$$

Proof (Theorem 2.6.17): Sec. A.2.3. In the proof, we use the symmetric KL divergence, rather than non-symmetric, since it lets us cancel out the partition functions of P, Q. Also, for the canonical parameterization for P, we take the expectation over reference assignment w.r.t. Q; and vice versa.

In the above theorem, let P be the target distribution and Q be our estimate. Then the theorem decomposes the error (symmetric KL) of our estimate into a set of local error terms of the form $\mathbf{E}\left[\log P_i - \log Q_i\right]$. If we could bound these local errors, then we could bound the error in the estimate of the full distribution. The first of the four types of local error terms in Eq. (2.57) is a local KL divergence: $KL(P_i \| Q_i) \doteq \mathbf{E}_{P(Y_i, Y_{N_i})}\left[\log P_i - \log Q_i\right]$; existing PAC-style bounds for regression may be used to bound this error term. However, the other error terms are non-standard. For example, the local KL divergence is always non-negative, but the other terms can be negative, so they are not clearly interpretable as error measures. We currently do not know how to bound them usefully.

Although we have not completed this last step in the analysis, we believe that Theorem 2.6.17 could improve upon the results of Abbeel et al. [2006] by permitting weaker assumptions. In particular, Abbeel et al. [2006] require uniformly good estimates of the local conditional distributions, as in Eq. (2.27). Each of the local error terms in Eq. (2.57) involves an expectation over Y_{N_i} or $Y_{N_i \setminus j}$ w.r.t. the target distribution P. We could thus use bounds on the local errors which involve these expectations w.r.t. P and permit higher error on parts of the distribution P which have low probability.

Theorem 2.6.17 may seem to suggest a new algorithm for parameter learning. In the theorem, if we let Q be our model and P be the target distribution, then the theorem decomposes the symmetric KL divergence into a set of local error terms for each variable Y_i. One might propose an algorithm which uses the right-hand side of Eq. (2.57) as a loss to be minimized w.r.t. the model Q. (This loss would be convex since the symmetric KL is convex.) The problem with this approach is that the local error terms involve expectations over $Q(Y_i)$, $\forall i$, and $Q(Y_i, Y_j)$, $\forall i, j \in Edges$. Computing these expectations requires inference over the full model Q. Our interest in the canonical parameterization was to develop a decomposable learning method which avoids inference over the full model; one might argue that, if we commit ourselves to performing inference, then we might as well optimize the symmetric KL (or non-symmetric KL) directly.

2.7 Discussion

Using pseudolikelihood (MPLE) and composite likelihood (MCLE), we proved the first finite sample complexity bounds for learning parameters of general MRFs and CRFs. For certain classes of models, these bounds constitute PAC bounds as well. Our bounds are written in terms of problem-specific constants, and through empirical analysis, we showed that these constants accurately determine the relative statistical efficiency of MLE, MPLE, and MCLE. Our small-scale tests give guidance for choosing MCLE structure in practice, even when our bounds' constants may not be computed.

We also helped to elucidate the relation between MPLE and the canonical parameterization—the one previous method capable of PAC-learning a general class of high-treewidth models. Our algorithmic improvements have moved the method one step closer to practicality, and our theoretical observations indicate the potential for usefulness in the future.

Most importantly, our work has demonstrated the potential for optimizing the trade-offs between sample complexity, computational complexity, and potential for parallelization when using MCLE for parameter learning. At one extreme, MLE has essentially optimal sample complexity, but MLE can have very high computational complexity and be difficult to parallelize. At the other extreme, MPLE has relatively high sample complexity, but low computational complexity and very simple parallelization. Ranging between MLE and MPLE is MCLE, which offers a very flexible class of estimators. On many models, smart choices of MCLE component structures can achieve sample complexity close to MLE, computational complexity close to MPLE, and simple parallelization. Moreover, the potential for joint or disjoint optimization offers another method for tuning these trade-offs. By tailoring these choices to our model and data, we can optimize these trade-offs and make learning parameters much more scalable.

2.8 Future Work

We divide our future work into three areas: (a) ideas stemming from our work on pseudolikelihood and composite likelihood, (b) extensions to our work on the canonical parameterization, and (c) extensions of our learning methods to alternate learning settings.

2.8.1 Pseudolikelihood and Composite Likelihood

In this section, we mention future work specific to pseudolikelihood and composite likelihood. We discuss much broader ideas in Ch. 6.

Automated choice of MCLE structure: In our examples, we choose the structure of our composite likelihood estimators by hand, using our expert knowledge about the structure of the graph and the correlations between random variables. It should be possible to choose structures automatically according to both the graph structure (which is given) and the correlations (which may be inferred from data). Choosing a structure could be phrased as graph partitioning, with the constraint that each partition have low treewidth. Using the graph structure would be most challenging on natural graphs with high-degree nodes, high connectivity (and low diameter), and irregular structure. Using correlations would be particularly interesting since estimating correlations would require a model of the distribution; thus, a good option might be a bootstrapped approach which uses a rough estimate of the distribution to choose a good estimator, which is in turn used to re-estimate the distribution.

Partly disjoint optimization: As parallel computing becomes more mainstream, more analysis of disjoint optimization could prove valuable. Currently, our analysis considers completely joint or disjoint optimization; of the two methods, joint optimization achieves lower sample complexity, while disjoint optimization is much easier to parallelize. Analyzing intermediate options, with limited communication between otherwise disjoint subproblems, might allow us to optimize this sample complexity–parallelization trade-off. A small amount of communication would not limit parallelism much but might greatly reduce sample complexity (i.e., improve accuracy for a given sample size). An approach using the Alternating Direction Method of Multipliers (ADMM) might work well. (For ADMM, see generally Boyd et al. [2011a].)

Combined analysis of learning and inference: We analyzed learning with the goal of recovering parameters. However, in practice, users are generally most interested in the performance of the model when answering test-time queries, i.e., running inference. Some work [Wainwright, 2006] has indicated that the same type of inference should be used during both learning and test-time. It is currently unclear what type of inference would be optimal with pseudolikelihood and composite likelihood, though it is easy to draw intuitive parallels with Gibbs sampling and structured Gibbs sampling, respectively [Asuncion et al., 2010]. A joint analysis of learning and inference might inform us of the optimal method for inference.

Dependence of bounds on eigenspectra; Analyzing non-identifiable models: Our MLE bounds currently depend on the minimum non-zero eigenvalue of the Hessian (of the loss at the target parameters). The use of the minimum is a worst-case choice which helps with our analysis. It might be possible to improve our analysis to depend on some combination of all eigenvalues; intuitively, low curvature of the objective in one direction (indicated by a small minimum eigenvalue) should only hurt our parameter estimates in that direction, not in other directions (corresponding to the other, larger eigenvalues). Such an analysis might also let us prove results for non-identifiable models.

Behavior of eigenspectra of various models: For small models, we can explicitly compute the eigenvalues required by our bounds. However, general rules dictating the behavior of the eigenvalues w.r.t. model structure, size, and parameters and w.r.t. estimator structure would be invaluable. Such rules would aid in automatically choosing estimator structures for new models and in proving PAC learnability of model classes.

Theoretical comparison with other methods: The relationships between many of learning methods discussed in Sec. 2.1 are unclear. For those methods which have consistency results but not finite sample complexity results, an analysis similar to ours might be possible, and it would aid in understanding these methods' relative performance.

Empirical comparison with other methods: In this work, we have focused on comparing MLE with MPLE and MCLE, rather than proving that MPLE and MCLE are better than any other learning method. Many authors have already proven MPLE and MCLE to be empirically competitive with other methods. However, since we have been able to show that carefully chosen MCLE structures can significantly outperform naively chosen structures, a broad comparison of carefully structured MCLE with additional learning methods would be valuable.

Model misspecification: Our analysis assumes that the model is well-specified. Generalizing our work to misspecified models would be valuable for understanding how to choose MCLE structures in real-world applications, in which we often cannot assume that the model is well-specified.

2.8.2 Canonical Parameterization

In this section, we mention future work specific to the canonical parameterization. We discuss much broader ideas in Ch. 6.

Sample complexity bounds: As discussed in Sec. 2.6.7, the canonical parameterization might be useful for proving error bounds (or sample complexity bounds) for learning methods such as pseudolikelihood. It is not clear how these bounds would compare to those developed in Sec. 2.3, but we suspect the bounds would look very different and might shed more light on the performance of pseudolikelihood on different types of models.

Algorithmic improvements: We proved that the canonical parameterization generalizes pseudolikelihood (MPLE) in a sense: certain algorithmic choices make the canonical parameterization equivalent to MPLE. The key algorithmic choices are how local conditional probabilities are estimated and how the estimates are averaged together to compute a single estimate of the full distribution. Other algorithmic choices could potentially improve upon MPLE and would be interesting to explore. (However, we note that our empirical tests have not yet revealed cases in which the canonical parameterization can outperform MPLE.)

Improving the canonical parameterization: If the canonical parameterization proves useful for sample complexity bounds (or an algorithm), it could be further improved. Our current formulation of the canonical parameterization handles large factors (factors involving many $\mathcal{Y}$ variables) inefficiently. A factor $\phi(Y_{C_j}, X_{D_j})$ in the true structure generates one canonical factor for every subset of C_j. With a single large factor, it is simple to show that all but two of these exponentially many canonical factors cancel out, greatly simplifying the canonical parameterization. It might be possible to use careful counting arguments to eliminate this exponential blowup in more general cases.

Relation to other methods: Sec. 2.6.5 compared the canonical parameterization with pseudolikelihood but left some open questions. Does the canonical parameterization benefit from comparing vs. a reference assignment? Are there classes of problems in which it will generally outperform pseudolikelihood? Also, in terms of held-out data log likelihood (in the tests discussed in Sec. 2.6.6), the canonical parameterization behaves more similarly to pseudolikelihood when we average over reference assignments and over $i_j \in C_j^*$; is it simply that averaging improves the performance of the canonical parameterization, or does averaging somehow make it more similar to pseudolikelihood? It would also be interesting to compare with other learning methods, such as piecewise likelihood [Sutton and McCallum, 2005, 2007], score matching [Hyvärinen, 2005], and ratio matching [Hyvärinen, 2007].

Model misspecification: Both Abbeel et al. [2006] and our work heavily rely on knowledge of the true structure, though Abbeel et al. [2006] do give bounds which permit partial misspecification. It would be interesting to explore empirically how the performance of the canonical parameterization degrades with model misspecification. Also, Liang and Jordan [2008] showed that the max-pseudolikelihood estimate degrades with model misspecification faster than the max-likelihood estimate; it would be interesting if the canonical parameterization could be used to shed light on this behavior.

2.8.3 Alternate Learning Settings

So far, we have discussed methods for scaling learning of *parametric* CRFs in the *supervised* setting, in which all variable values are fully observed in the training data. In this section, we propose extensions of our work to other settings. The first proposal, nonparametric CRFs, suggests a path for extending the work to more flexible model representations. The second proposal, semisupervised learning, discusses an extension to the setting in which some variable values are unobserved.

Nonparametric CRFs

Nonparametric models can permit more flexibility than parametric representations. Rather than assuming a parametric form of a distribution, such models learn a data-dependent representation. However, the extra computation generally required by nonparametric learning can make nonparametric CRFs prohibitively expensive to learn. We propose limiting this expense by applying pseudolikelihood and composite likelihood (Ch. 2) to nonparametric models.

Song et al. [2009, 2010] developed Hilbert space embeddings for constructing and operating on nonparametric representations of marginal and conditional distributions. They demonstrate empirically that the flexibility of nonparametrics permits the learning of more accurate models when the data do not come from a simple parametric distribution.

The methods from Song et al. [2009, 2010] may be generalized to CRFs and used in MLE training, but the operations required can be very expensive (though technically polytime) since MLE training requires running inference many times. We propose to use their nonparametric representations for pseudolikelihood and composite likelihood. A particular challenge is finding a nonparametric representation which permits simple estimation of pseudolikelihood or composite likelihood components from data, as well as factor averaging to create a complete model which takes the appropriate nonparametric form.

Nonparametric CRFs have been previously proposed by Lafferty et al. [2004], under the title kernel CRFs. We propose to use the methods from Song et al. [2009, 2010] since the methods from Lafferty et al. [2004] implicitly require that the output variables $\mathcal{Y}$ be discrete-valued.

Semi-supervised Learning

We propose extending our ideas for composite likelihood (and pseudolikelihood) to the semi-supervised setting, in which variables may be partially observed. Semi-supervised learning typically requires more computation than supervised learning, and semi-supervised methods are often posed as global optimization problems over the full model (e.g., posterior regularization [K. Ganchev and Taskar, 2010] and generalized expectation [McCallum et al., 2007]). Decomposing the objective might therefore greatly improve scalability. However, composite likelihood itself may not be directly applicable since its subproblems involve estimating conditional probabilities; when the variables being conditioned on are only partially observed, it can be difficult to solve such subproblems in isolation.

Chapter 3

CRF Structure Learning

As discussed in Sec. 1.3, research on CRF structure learning is still in a nascent stage, and even recovering tree structures is an unsolved problem. In this chapter, we discuss initial work exploring this problem of learning tree structures for CRFs. We begin with tree structures partly for their simplicity and partly since they permit exact inference at test time. As Shahaf et al. [2009] demonstrate, low-treewidth structures can produce better results than more general structures which require approximate inference.

Our work centers around generalizing the Chow-Liu algorithm for tree MRFs [Chow and Liu, 1968] to CRFs. We use the same algorithmic framework: compute edge weights, and choose a maximum-weight spanning tree. The framework is simple, efficient, and easy to parallelize since the main expense is in computing the edge weights. We are most interested in settings with large numbers of input variables, for it is in these settings that the obvious generalization of Chow-Liu breaks down (as shown in Sec. 3.2.1). To cope with large numbers of input variables, we take advantage of the idea of *evidence locality*, i.e., local structure w.r.t. input variables, where only a small subset of input variables in $\mathcal{X}$ participate in each factor.

Our results so far are mixed: we present one major negative theoretical result (in Sec. 3.2.2) but encouraging empirical results (in Sec. 3.4). In Sec. 3.6, we propose several improvements which might help us sidestep around our negative result.

Our main contributions in this chapter may be summarized as follows:

- Analysis of conditions for recovering tree CRF structures
- Generalization of Chow-Liu for CRFs, via generalized edge weights
    - Negative theoretical result for the generalized edge weights
    - Discussion of select edge weights with intuitive motivations
- Empirical tests of select edge weights on synthetic and fMRI data, with good performance

This chapter is based on work initially presented in Bradley and Guestrin [2010].

3.1 Related Work

We drew inspiration from several sources. Our work is mostly based upon the Chow-Liu algorithm from Chow and Liu [1968]. Related methods are used by Friedman et al. [1997] to learn Tree-Augmented Naive Bayes classifiers and by Shahaf et al. [2009] to learn low-treewidth CRFs; we discuss these works in Sec. 3.2.1. One of the edge weights we consider (in Sec. 3.3.1) is based on piecewise likelihood [Sutton and McCallum, 2005].

Concerning structure learning more generally, several heuristic methods have been empirically successful. Torralba et al. [2004] proposed Boosted Random Fields, a method which selects features and structure using greedy steps to approximately maximize data log likelihood. Schmidt et al. [2008] proposed maximizing block-ℓ_1-regularized pseudolikelihood, which gives a convex program and tends to produce sparse models. Both methods learn high-treewidth models in general.

As mentioned in Sec. 1.3, structure learning is known to be hard in general [Srebro, 2003]. One of the few learnability results we are aware of comes from Ravikumar et al. [2010], whose results on learning Ising MRF structures imply learnability for some classes of Ising CRFs. They use L_1-regularized regression to discover each variable's neighbors. Lee et al. [2006] take a similar approach, using L_1-regularized log likelihood to learn high-treewidth MRF structures. However, the method from Lee et al. [2006] requires intractable inference (or approximations without guarantees) during learning since the objective includes the full likelihood.

Chechetka and Guestrin [2010] proposed the interesting idea of evidence-specific structures for CRFs, in which a new structure over $\mathcal{Y}$ is chosen for each test example (y, x) based on learned correlations in $P(\mathcal{Y}|x)$. Though their method lacks theoretical guarantees, it can improve upon traditional CRFs in practice. They use a similar Chow-Liu-based algorithmic approach for choosing structures, but they use global Conditional Mutual Information for edge weights; as we discuss in Sec. 3.2.1, this choice is not ideal when $|\mathcal{X}|$ is large. However, our alternative edge weights may be plugged directly into their method.

3.2 Efficiently Recovering Tree CRFs

In this chapter, we use the notation for CRFs defined in Ch. 1. We assume we are given an *input mapping* specifying inputs X_{D_j} for each possible factor involving Y_{C_j}. (We briefly discuss methods for feature selection in Sec. 3.3.5 and give empirical evidence of their efficacy.) For many applications, expert knowledge can provide input mappings (such as in image segmentation, where $\mathcal{Y}$ are segment labels and $\mathcal{X}$ are pixels); in other cases, sparsistent methods [Ravikumar et al., 2008] can be used for feature selection. Like us, Schmidt et al. [2008] and Shahaf et al. [2009] assume pre-specified input mappings.

Since we only learn tree CRFs, we simplify notation. We write $Y_{ij} \equiv \{Y_i, Y_j\}$. The inputs for Y_i specified by the input mapping are X_i; likewise, X_{ij} corresponds to Y_{ij}. We index factors similarly: $\psi_i = \psi_i(Y_i, X_i)$ and $\psi_{ij} = \psi_{ij}(Y_{ij}, X_{ij})$. Our goal now is a scalable algorithm for learning tree CRF structures. We begin by proposing a gold standard and showing it is ideal but impractical.

Algorithm 3.1: Tree CRF Structure Learner Template

```
Input: Dataset $D$ over $\mathcal{Y}, \mathcal{X}$; inputs $X_i$ for each $Y_i$
Initialize $G$ to be the complete graph over $\mathcal{Y}$.
foreach $(Y_i, Y_j)$ do
    In $G$, set $Weight(Y_i, Y_j) \longleftarrow Score(i, j)$.
return $MaxSpanningTree(G)$
```

3.2.1 A Gold Standard

We can define an algorithm analogous to Chow-Liu for generative models [Chow and Liu, 1968] by showing that the conditional log likelihood of a tree CRF *decomposes* over the edges (i, j) and vertices i in the tree T. Let Q be our model and P the true distribution. Using the (optimal) parameters $Q(A|B) = P(A|B)$,

$$\begin{aligned} \mathbf{E}_P\left[\log Q(\mathcal{Y}|\mathcal{X})\right] &= \textstyle\sum_{(i,j)\in T} \mathbf{E}_P\left[\log Q(Y_{ij}|\mathcal{X})\right] \\ &\quad - \textstyle\sum_i (deg_i - 1)\mathbf{E}_P\left[\log Q(Y_i|\mathcal{X})\right] \\ &= \textstyle\sum_{(i,j)\in T} I_P(Y_i; Y_j|\mathcal{X}) + C, \end{aligned}$$

where subscript P means w.r.t. P, deg_i is the degree of vertex i, and C is a constant w.r.t. the structure.

Assuming the *global Conditional Mutual Information (CMI)* $I(Y_i; Y_j|\mathcal{X})$ is easy to compute, we can recover the maximum likelihood model: weight each possible edge (i, j) with $I(Y_i; Y_j|\mathcal{X})$, and choose the maximum-weight spanning tree (MST). This method was proposed by Friedman et al. [1997] for learning Tree-Augmented Naive Bayes classifiers. Unfortunately, as the dimensionality of $\mathcal{X}$ grows, this method quickly becomes intractable. Computing $I(Y_i; Y_j|\mathcal{X})$ requires an accurate estimate of $P(Y_{ij}|\mathcal{X})$ (or similar quantities); $P(Y_{ij}|\mathcal{X})$ can be as expensive to compute and represent as $P(\mathcal{Y}|\mathcal{X})$ when the dimensionalities of $\mathcal{Y}$ and $\mathcal{X}$ are of the same order. This observation emphasizes the need to parameterize our model with *local inputs* $X_i \subset \mathcal{X}$, rather than *global inputs* $X_i = \mathcal{X}$, to ensure scalability w.r.t. $\mathcal{X}$. [1]

Ideally, we could retain the efficiency of spanning trees while making use of local inputs in $\mathcal{X}$, i.e., only calculating probabilities of the form $P(Y_{ij}|X_{ij})$ conditioned on small sets X_{ij}. With local inputs, though, the partition function prevents the conditional log probability from decomposing over edges and vertices:

$$\log P(\mathcal{Y}|\mathcal{X}) = -\log Z(\mathcal{X}) + \sum_{(i,j)\in T} \log \psi_{ij}(Y_{ij}, X_{ij}).$$

Our primary goal is to overcome the intractability of the partition function by deriving a meaningful local edge score $Score(i, j)$ usable in Algorithm 3.1.

Shahaf et al. [2009] took a similar approach, defining edge scores and maximizing the weight of edges chosen for a low-treewidth model. However, they used global inputs, with edge scores set to the global CMI $I(Y_i; Y_j|\mathcal{X})$. They focused on the second step—maximizing the weight of edges included in the model—while we focus on better methods for weighting edges. Our work is compatible with theirs; their algorithm could use our edge scores to learn treewidth-k models.

[1] If the true model were a tree CRF with local inputs and complexity were ignored, global CMI could recover the true structure, after which the model could be succinctly parameterized with local inputs. However, if the true CRF were not a tree, it is unclear whether global CMI would recover the optimal projection onto a tree with local inputs.

3.2.2 Score Decay Assumption

We phrase our analysis of edge scores in terms of recovering the structure of tree CRFs. I.e., we assume the true distribution is representable by a tree CRF. We begin by defining a desirable property for edge scores.

Definition: If the true model $P(\mathcal{Y}|\mathcal{X})$ is a tree T, the *Score Decay Assumption (SDA)* states that, for any edge $(i,j) \in T$, if a path in T of length > 1 between k,l includes (i,j), then $Score(i,j) > Score(k,l)$.

Intuitively, the SDA says the score between vertex pairs decays with distance. Yet this is less strict than requiring, e.g., that the assumption hold regardless of whether (i,j) is an edge in T; thus, the SDA does not rely on comparing pairs of edges not in T. This condition is necessary and sufficient for recovering trees.

Theorem 3.2.1. *Suppose $P(\mathcal{Y}|\mathcal{X})$ is representable by a tree CRF with structure T. The Score Decay Assumption holds for a score S w.r.t. P iff Algorithm 3.1 using score S can recover T.*

Proof: While building a maximum spanning tree over $\mathcal{Y}$ by Kruskal's Algorithm [Kruskal, Jr., 1956], say we add edge $(k,l) \notin T$. Let $path_{kl}$ be the path between k,l in T. There exists an edge $(i,j) \in path_{kl}$ not yet added, so $Score(i,j) < Score(k,l)$, violating the SDA. I.e., we add an edge not in T iff the SDA is violated, so Algorithm 3.1 recovers T iff the SDA holds. ■

Recall that we wish to use local scores $S(i,j) = f(Y_{ij}, X_{ij})$ for scalability. We make a simplifying assumption about the input mapping: a factor involving Y_{ij} depends on $X_i \cup X_j$ (not arbitrary X_{ij}). This assumption makes our methods less general but more practical. (E.g., feature selection requires inputs for each of $|\mathcal{Y}|$ outputs, rather than for $\binom{|\mathcal{Y}|}{2}$ possible edge factors.) However, we note that our approach still permits the use of global inputs, i.e., inputs which are included in all factors. Schmidt et al. [2008] make use of a mix of global and local inputs, while Shahaf et al. [2009] use only global inputs. We now define a general class of local scores.

Definition: The *Local Linear Entropy Scores* are scores $Score(i,j)$ representable as linear combinations of entropies over subsets of $\{Y_i, Y_j, X_i, X_j\}$. (The entropy of a distribution $P(X)$ is defined as $H(X) \doteq \mathbf{E}_{P(X)}\left[-\log P(X)\right]$. The conditional entropy of a distribution $P(Y|X)$ (when there is an implicit corresponding distribution $P(X)$) is defined as $H(Y|X) \doteq \mathbf{E}_{P(X)}\left[\mathbf{E}_{P(Y|X)}\left[-\log P(Y|X)\right]\right]$.)

This class of scores includes, for example, the local Conditional Mutual Information $I(Y_i;Y_j|X_{ij}) = H(Y_i|X_{ij}) + H(Y_j|X_{ij}) - H(Y_{ij}|X_{ij})$ we consider in Sec. 3.3.2. Unfortunately, the Local Linear Entropy Scores are insufficient in general for recovering tree CRFs, as the following theorem demonstrates.

Theorem 3.2.2. *Assume that the edge score S is symmetric, i.e., $S(i,j) = S(j,i)$. Even if we assume the class of distributions we are learning:*

- *are representable by tree CRFs,*
- *obey the input mapping, i.e., that if the true model has edge (i,j), a factor involving Y_{ij} involves no more inputs than X_{ij},*
- *and have no trivial factors, i.e., that no factors are deterministic or effectively absent,*

then every Local Linear Entropy Score S violates the Score Decay Assumption for some models from this class, even with exact entropy estimates.

Proof Sketch: Since conditional entropies equal a difference between non-conditional entropies, any Local Linear Entropy Score $S(i,j)$ may be written as

$$\begin{aligned}\mathbf{w}\cdot\big(&H(Y_i)+H(Y_j),H(X_i)+H(X_j),H(Y_{ij}),H(X_{ij}),\\&H(Y_i,X_i)+H(Y_j,X_j),H(Y_i,X_j)+H(Y_j,X_i),\\&H(Y_i,X_{ij})+H(Y_j,X_{ij}),H(Y_{ij},X_i)+H(Y_{ij},X_j),\\&H(Y_{ij},X_{ij})\big)\end{aligned}$$

(This has all non-conditional entropies, grouped since S is symmetric.) The proof considers a list of cases using these observations: **(1)** Since tree CRFs generalize trees (where $\mathcal{X}=\emptyset$), the score must (approximately) reduce to the mutual information $S(i,j)=I(Y_i;Y_j)$ when $\mathcal{X}=\emptyset$. **(2)** Since we can have arbitrary factors over $\mathcal{X}$, the score must not be exactly $S(i,j)=I(Y_i;Y_j)$. **(3)** We can introduce simplifying constraints by considering models for which the inputs $\mathcal{X}$ are conditionally independent given the outputs $\mathcal{Y}$.
These constraints allow us to prove that, for certain classes of distributions, we require $S(i,j)\approx I(Y_i;Y_j)$, but that such a score fails for other classes. ■

3.3 Heuristic Scores

Despite this negative result, we are able to identify certain Local Linear Entropy Scores which are intuitive and have desirable theoretical properties. Though the scores violate the Score Decay Assumption in general, they are very successful empirically (Sec. 3.4).

3.3.1 Piecewise Likelihood

We want to base our scores upon the conditional log likelihood, which does not decompose over edges because of the partition function $Z(\mathcal{X})$. Much research aims at handling partition functions tractably. We frame our analysis in terms of one such work: piecewise likelihood [Sutton and McCallum, 2005, 2007].

Piecewise likelihood approximates the log likelihood by upper-bounding $Z(\mathcal{X})$. The bound effectively divides $Z(\mathcal{X})$ into one term for each factor, permitting learning each factor's parameters independently. Specifically, the piecewise likelihood objective ℓ_{PW} is

$$\ell_{PW}(\mathcal{Y}|\mathcal{X})=\sum_j \log\psi_j(Y_j,X_j)-Z_j(X_j)\,, \tag{3.1}$$

where $Z_j(X_j)$ is a local partition function which normalizes $P_j(Y_j|X_j)\propto\psi_j(Y_j,X_j)$ to be a distribution. Sutton and McCallum [2005] prove that $\log P(\mathcal{Y}|\mathcal{X})\geq\ell_{PW}(\mathcal{Y}|\mathcal{X})$, so maximizing piecewise likelihood maximizes a lower bound on the log likelihood. Sutton and McCallum [2005, 2007] apply this approximation to parameter learning for Markov random fields and CRFs, achieving fast learning and high accuracy. For pairwise models, if we combine factors with their Z terms, piecewise likelihood becomes $\sum_{(i,j)}\log P(Y_{ij}|X_{ij})$, where this local conditional distribution is modeled using only the factors whose domains are in $Y_{ij}\cup X_{ij}$. This Local Linear Entropy Score is usable in Alg. 3.1: $S(i,j)=\mathbf{E}_P\left[\log P(Y_{ij}|X_{ij})\right]$.

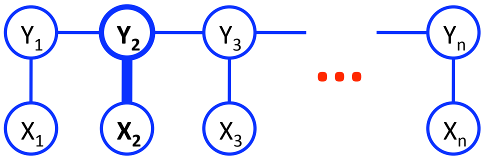


Figure 3.1: Counterexample for piecewise likelihood and local CMI: $P(\mathcal{Y}, \mathcal{X})$ has a comb structure, with weak pairwise factors everywhere except for one strong factor $\psi(Y_2, X_2)$.

However, piecewise likelihood is a poor edge score. If a pair (Y_i, X_i) share a strong factor, piecewise likelihood weights will likely result in a star structure over $\mathcal{Y}$ with Y_i at the center, as in the following example.

Example 1: Consider Figure 3.1's model. (Y_2, X_2) has a strong factor so that $H(Y_2|X_2) \approx 0$. The score for any edge $(2, i)$ is:

$$\begin{aligned} \mathbf{E}\left[\log P(Y_{2,i}|X_{2,i})\right] &= -H(Y_{2,i}|X_{2,i}) \\ &= -H(Y_i|X_{2,i}) - H(Y_2|Y_i, X_{2,i}) \\ &\approx -H(Y_i|X_{2,i}) > -H(Y_i|X_i) \end{aligned}$$

All other X_j participate in weak factors, so $-H(Y_i|X_i) \approx -H(Y_i|X_{ij}) > -H(Y_{ij}|X_{ij})$. Even if edge (Y_2, Y_i) is not in the true model and (Y_i, Y_j) is, we will score (Y_2, Y_i) above (Y_i, Y_j). This behavior appeared often in our synthetic experiments. ■

Nevertheless, piecewise likelihood is a useful approximation which helps tie our two proposed edge scores to the ideal but intractable global CMI.

3.3.2 Local CMI

The first Local Linear Entropy Score we select is a local version of global CMI $I(Y_i; Y_j|\mathcal{X})$. The *local CMI* is:

$$I(Y_i; Y_j|X_{ij}) = \mathbf{E}_P\left[\log P(Y_{ij}|X_{ij}) - \log P(Y_i|X_{i,j}) - \log P(Y_j|X_{i,j})\right].$$

The local CMI score is interpretable as the piecewise likelihood (first term in the expectation), minus correction terms for the edge's endpoints (second and third terms in the expectation). Intuitively, these corrections discount interactions between each Y_i and $\mathcal{X}$. We can also interpret local CMI as a bound on the likelihood gain of a tree CRF over a disconnected model.

Proposition 3.3.1. *Let $Q_T(\mathcal{Y}|\mathcal{X})$ be the projection of the true distribution $P(\mathcal{Y}|\mathcal{X})$ onto tree structure T w.r.t. $P(\mathcal{X})$, and let $Q_{disc}(\mathcal{Y}|\mathcal{X}) \equiv \prod_i P(Y_i|X_i)$ be the projection onto the disconnected model. The local CMI score $CMI(i,j)$ for Q_T bounds the likelihood gain* [2]*:*

$$\mathbf{E}_P\left[\log Q_T(\mathcal{Y}|\mathcal{X}) - \log Q_{disc}(\mathcal{Y}|\mathcal{X})\right] \geq \sum_{(i,j)\in T} CMI(i,j)$$

[2] Both piecewise likelihood and local CMI have bounds relating them to the tree CRF log likelihood; we can show that neither bound is strictly better for all distributions.

Proof: Choose a topological ordering of $\mathcal{Y}$ in T with root Y_1. Let Y_{Pa_i} be Y_i's parent. Entropies and expectations are defined w.r.t. $Q'(Y, X) \equiv Q_T(\mathcal{Y}|\mathcal{X})P(\mathcal{X})$.

$$\begin{aligned}
&E[\log Q_T(\mathcal{Y}|\mathcal{X}) - \log Q_{disc}(\mathcal{Y}|\mathcal{X})] \\
&= \textstyle\sum_i H(Y_i|X_i) - H(\mathcal{Y}|\mathcal{X}) \\
&= \textstyle\sum_i H(Y_i|X_i) - H(Y_1|X) - \sum_{i>1} H(Y_i|Y_{Pa_i}, X) \\
&= H(Y_1|X_1) - H(Y_1|X) + \textstyle\sum_{i>1} H(Y_i|X_i) - H(Y_i|Y_{Pa_i}, X) \\
&\geq H(Y_1|X_1) - H(Y_1|X_1) + \textstyle\sum_{i>1} H(Y_i|X_{i,Pa_i}) - H(Y_i|Y_{Pa_i}, X_{i,Pa_i}) \\
&= \textstyle\sum_{i>1} H(Y_i|X_{i,Pa_i}) + H(Y_{Pa_i}|X_{i,Pa_i}) - H(Y_{i,Pa_i}|X_{i,Pa_i}) \\
&= \textstyle\sum_{(i,j)\in T} CMI(i,j) \quad \blacksquare
\end{aligned}$$

Like piecewise likelihood, local CMI can perform poorly if a pair (Y_i, X_i) has a strong factor.

Example 2: Consider again Figure 3.1's model, where $\psi(Y_2, X_2)$ is strong enough that $H(Y_2|X_2)$ is small. Local CMI gives a small score to any edge with Y_2 since

$$I(Y_2; Y_j|X_{2,j}) = H(Y_2|X_{2,j}) - H(Y_2|Y_j, X_{2,j}) \approx 0.$$

Since the score for the false edge (Y_1, Y_n) does not condition on X_2, it could be much higher than the score of the true edges $(Y_1, Y_2), (Y_2, Y_3)$. Note that this is a separate issue from identifiability; with local inputs, it is vital that we connect Y_1, Y_2 and Y_2, Y_3. ■

However, local CMI performs fairly well in practice.

3.3.3 Decomposable Conditional Influence

To overcome the above counterexample, we propose a final Local Linear Entropy Score dubbed the Decomposable Conditional Influence (DCI):

$$DCI(i,j) \equiv \mathbf{E}_P \left[\log P(Y_{ij}|X_{ij}) - \log P(Y_i|X_i) - \log P(Y_j|X_j)\right].$$

The first term in the expectation is the piecewise likelihood (as for local CMI), but the second and third equal the edge's endpoints' scores in the disconnected model $P_{disc}(\mathcal{Y}|\mathcal{X}) \equiv \prod_i P(Y_i|X_i)$, giving the following result:

Proposition 3.3.2. *When building a spanning tree T, if we add edge (i, j) and T does not yet contain edges adjacent to i, j, then DCI is an exact measure of the likelihood gain from adding edge (i, j).*

Moreover, DCI succeeds on Figure 3.1's counterexample for piecewise likelihood and local CMI.

Example 2, continued: The DCI edge scores are:

$$\begin{aligned}
&DCI(1,2) \\
&\quad = -H(Y_{1,2}|X_{1,2}) + H(Y_1|X_1) + H(Y_2|X_2) \\
&\quad \approx -H(Y_1|X_{1,2}) + H(Y_1|X_1) \\
&DCI(1,n) \\
&\quad = -H(Y_{1,n}|X_{1,n}) + H(Y_1|X_1) + H(Y_n|X_n) \\
&\quad = [H(Y_1|X_1) - H(Y_1|X_{1,n})] + [H(Y_n|X_n) - H(Y_n|Y_1, X_{1,n})]
\end{aligned}$$

Since Y_1, Y_n are far apart, the two terms in $DCI(1, n)$ are much closer to 0 than the sum in $DCI(1, 2)$. ■

DCI performs very well in practice (Sec. 3.4.1).

3.3.4 Sample Complexity

Though Local Linear Entropy Scores violate the Score Decay Assumption in general, it is instructive to consider the sample complexity if the assumption is met.

Theorem 3.3.3. *Let $\mathcal{Y}, \mathcal{X}$ be discrete and $P(\mathcal{Y}|\mathcal{X})$ be representable by a CRF with tree structure T. (Assume w.l.o.g. that X_i is a single variable; we can merge multiple variables into a new variable of higher arity.) Let all variables have arity $\leq R$. Let $|\mathcal{Y}| = n$. Assume a Local Linear Entropy Score S meets the Score Decay Assumption by ϵ; i.e., for each edge $(i,j) \in T$ on the path between k, l, $S(i,j) - S(k,l) > \epsilon$.*
To recover the tree with probability at least $1-\gamma$, it suffices to train on a set of i.i.d. samples of size

$$O\left(\frac{R^8}{\epsilon^2}\log^2\left(\frac{R}{\epsilon}\right)\left(\log n + \log\frac{1}{\gamma} + \log R\right)\right)$$

Proof: We use this result on the sample complexity of estimating entropies from Höffgen [1993]: The entropy over k discrete variables with arity R may be estimated within absolute error Δ with probability $\geq 1-\gamma$ using $O\left(\frac{R^{2k}}{\Delta^2}\log^2\left(\frac{R^k}{\Delta}\right)\log\left(\frac{R^k}{\gamma}\right)\right)$ i.i.d. samples (and time).

Local Linear Entropy Scores may be represented by a constant number of entropies over $\leq k = 4$ variables. With $\binom{n}{2}$ possible edges, we must compute $O(n^2)$ entropies. To ensure estimates of scores order edges in the same way as the true scores, we must estimate scores within error $\Delta = \epsilon/2$. These values, Hoffgen's result, and a union bound complete the proof. ■

This theorem makes 2 key predictions: 1) sample complexity will increase only logarithmically in $n = |\mathcal{Y}|$, and 2) high arity R drastically increases sample complexity, indicating the importance of local inputs. Both predictions are born out by our empirical results.

3.3.5 Feature Selection

We have assumed we are given inputs X_i for each Y_i. If this mapping is unavailable, we recommend ℓ_1-regularized regression. Ravikumar et al. [2008] present sparsistent methods which, given certain assumptions, can recover the neighbors of variables in the model. Though their guarantees are asymptotic in sample size, they show the method works empirically.

To choose inputs for Y_i using ℓ_1-regularized regression, one could use two methods. The unconservative method (choosing larger sets X_i) regresses $Y_i \sim \mathcal{X}$. The conservative method regresses $Y_i \sim \mathcal{X} \cup (\mathcal{Y} \backslash Y_i)$. Sparsistency results indicate that, if $P(\mathcal{Y}|\mathcal{X})$ is representable by a tree CRF, the conservative method may be better. However, the conservative one might be too selective if the true structure is not a tree. In our fMRI experiments, we found computational complexity and relative performance favor the conservative method.

3.4 Experiments

We first use synthetic data to compare local CMI and DCI against other CRF learning methods: piecewise likelihood, global CMI, and block-ℓ_1-regularized pseudolikelihood [Schmidt et al., 2008]. (We omit

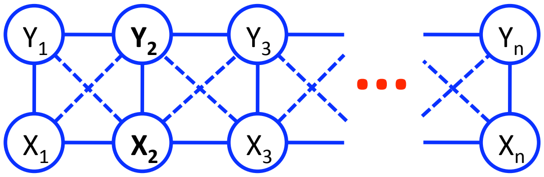


Figure 3.2: Tractable chain structure for $P(\mathcal{Y}, \mathcal{X})$ used in synthetic experiments. The dotted edges represent *cross-factors*. This model is tractable since the structures of $P(\mathcal{Y}|\mathcal{X})$ and $P(\mathcal{X})$ match: both are chains with corresponding variables Y_i, X_i in the same order.

results from piecewise likelihood since it does poorly.) We also tested two unstructured models: the *disconnected CRF with global inputs* $P(\mathcal{Y}|\mathcal{X}) = \prod_i P(Y_i|\mathcal{X})$ and the *disconnected CRF with local inputs* $P(\mathcal{Y}|\mathcal{X}) = \prod_i P(Y_i|X_i)$. Based on these results, we tested the best methods on the larger-scale fMRI application.

3.4.1 Synthetic Models

We tested a wide variety of synthetic models over binary variables; the models varied as follows:
Chains vs. trees: Chains have joint distributions $P(\mathcal{Y}, \mathcal{X})$ representable by ladders composed of cliques $(Y_i, Y_{i+1}), (X_i, X_{i+1}), (Y_i, X_i)$. Trees are the natural generalization, generated using non-preferential random attachment [Nakazatoa and Arita., 2007]. We tested with and without *cross factors* $\psi(Y_i, X_{i+1})$.
Tractable vs. intractable joint models $P(\mathcal{Y}, \mathcal{X})$: For our tractable models, $P(\mathcal{Y}, \mathcal{X})$ may be sampled from directly. For our intractable models, $P(\mathcal{X})$ and $P(\mathcal{Y}|\mathcal{X})$ are tractable, but $P(\mathcal{Y}, \mathcal{X})$ is not (but may be sampled via $x \sim P(\mathcal{X}), y \sim P(\mathcal{Y}|\mathcal{X} = x)$).
Associative vs. random factors: Associative factors set $\psi(A, B) = \exp(s)$ if $A = B$ and $\psi(A, B) = 1$ if $A \neq B$, where s is a factor strength. Random factors have each value $\log \psi(a, b)$ sampled from $Uniform[-s, s]$. We set strengths s separately for Y-Y, Y-X, and X-X factors. For associative factors, we tried both fixed and alternating positive and negative strengths.

These models have natural input mappings from Y_i to X_i which we gave our learners access to.

Exact Scores

We tested exact scores on simple models to illuminate domains in which local CMI and DCI succeed. Figure 3.3 shows model recovery results for length-10 chains. DCI outperforms local CMI when Y-X factors are stronger (matching the counterexample in Figure 3.1) and when Y-Y factors are weak. Peculiarly, local CMI does better when X-X factors are strong. These trends were similar for most models we tested, though local CMI did significantly better without alternating positive and negative factors.

Fig. 3.4 compares the magnitudes of the edge scores for local CMI and DCI for an example test. The median magnitude of DCI scores is about 10 times higher than that of local CMI. Since scores for these simple length-10 chains go as low as 10^{-8}, this difference could indicate more danger of numerical instability for local CMI.

Tests with Samples

Our next tests used samples from the synthetic models. For structure learning, we computed $P(Y_i, Y_j|X_C)$ where X_C is small using tables of counts. For large sets X_C, we used ℓ_2-regularized logistic regres-

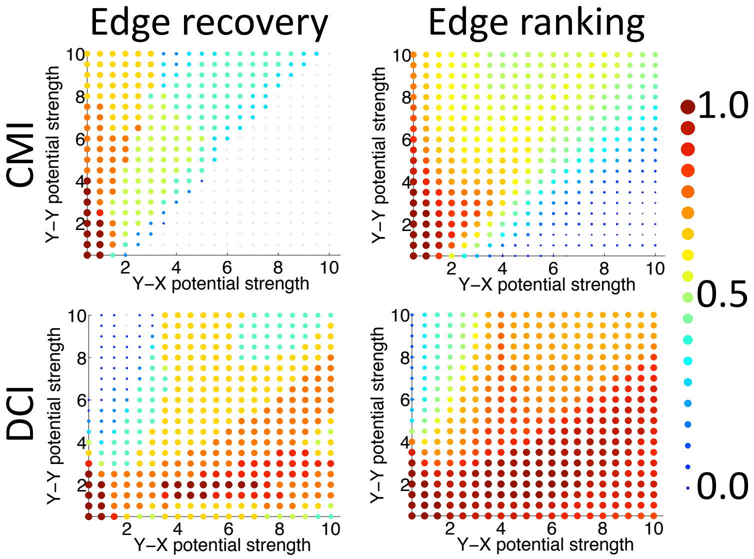


Figure 3.3: Local CMI vs. DCI with exact scores. Chains with associative factors, cross factors, alternating $+/-$ strengths. Plots: varying magnitudes of Y-Y and Y-X strengths for fixed magnitude 2 for X-X (in log space). "Edge recovery" is the fraction of true edges recovered; "edge ranking" is the fraction of (true, false) edge pairs with the true edge ranked above the false. $|\mathcal{Y}| = |\mathcal{X}| = 10$.

sion. We chose regularization parameters separately for every regression during structure learning, which outperforms fixed regularization. We smoothed estimates of $P(A|B = b)$ with one extra example per $a \in Val(A)$.

For parameter learning, we used conjugate gradient to maximize the ℓ_2-regularized data log likelihood. For both structure and parameter learning, we chose regularization parameters via 10-fold cross-validation, testing 10 values between .001 and 50 (on a log scale). Global CMI has a natural parameterization with factors $P(Y_i, Y_j|\mathcal{X})$, but to be fair, we switched to factors with local inputs $\psi(Y_i, Y_j, X_{ij})$ during parameter learning, which gave higher performance.

We tested the block-ℓ_1-regularized pseudolikelihood method from Schmidt et al. [2008] using their implementation of structure and parameter learning and inference (via loopy belief propagation).

Figures 3.5 and 3.6 show results for tree CRFs with intractable joints $P(\mathcal{Y}, \mathcal{X})$, associative factors with alternating factors (Y-Y, Y-X, X-X alternating between ±4, ±2, ±1, respectively, in log space), with cross factors. "Test accuracy" is 0/1 (predicting all of $\mathcal{Y}$ or not). Training time includes cross-validation for choosing regularization for structure but not parameter learning. Figure 3.5 compares varying training set sizes, while Figure 3.6 varies the model size.

In both, DCI consistently outperforms other methods in recovering true edges, except for small sample sizes. Global CMI is the next most competitive, overtaking DCI in accuracy with enough training data. However, global CMI becomes prohibitively expensive as the training set and model sizes increase. Like DCI, local CMI is tractable, but it under-performs DCI.

These tests are difficult for the Schmidt et al. [2008] method, for it learns general CRFs, not tree CRFs. The plots omit log likelihood for Schmidt et al. since it is intractable to compute, though it could be approximated via a projection. We omit results from other models for lack of space. In general, DCI

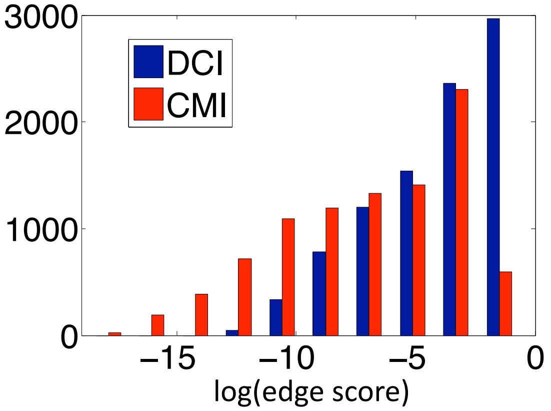


Figure 3.4: Histogram of the edge score magnitudes of local CMI and DCI for one of the tests in Figure 3.3. The x-axis uses log with base e. The relative magnitudes of the two score types indicate that local CMI may be more likely to suffer from numerical instability than DCI.

performs best, especially with large models and random factors.

Though local CMI and DCI do not obey the Score Decay Assumption in general, we observed that they approximately follow it. Figure 3.7 plots SDA violation for consecutive triplets (i, j, k) in the true CRF, measured as $(1/2)[(S(i,k) - S(i,j)) + (S(i,k) - S(j,k))]$. SDA violation and edge recovery are strongly anti-correlated.

3.4.2 fMRI

We next applied our CRF learning methods to an fMRI application from Palatucci et al. [2009]. The learners takes inputs $\mathcal{X}$ which are voxels (3-D pixels) from fMRI images of test subjects' brains and predicts a vector $\mathcal{Y}$ of semantic features which describe what the test subject is thinking of (e.g., "Is it man-made?"; "Can you hold it?"). This application is much more challenging than our synthetic experiments. After pre-processing, their dataset has 60 examples (objects), with $|\mathcal{Y}| = 218$ and $|\mathcal{X}| = 500$, for each of 9 test subjects. Palatucci et al. [2009] gives more details.

Given the success of DCI in synthetic tests, we chose it for the fMRI data. $\mathcal{Y}$ and $\mathcal{X}$ are real-valued, so we used conditional Gaussian factors $\psi(y, x) = \exp(-(1/2)(Ay - (Cx + b))^2$, where y, x are vectors. This parameterization is similar to that of Tappen et al. [2007], though they do not do general parameter learning, and it permits unconstrained optimization. [3] We regularized A and C, b separately, choosing regularization via 10-fold cross validation (CV) on values between .0001 and 30 (in a grid in log space). Because of the expense of CV, we ran CV on test subject 0 and used the chosen regularization for subjects 1-8.

With no natural input mapping, we used ℓ_1-regularized regression to do feature selection. To decrease the number of parameters (for both computational and statistical benefits), we tried two methods: **CRF 1**: We chose ≤ 10 highest-weight inputs per Y_i, accounting for 1/5 of the regression weights on average. To use all of $\mathcal{X}$ without increasing complexity, we added fixed factors $\psi(Y_i, \mathcal{X}) = P(Y_i|\mathcal{X})$, $\forall i$.

[3] We technically must constrain A so that $A^T A$ is invertible, but this was not a problem in our tests.

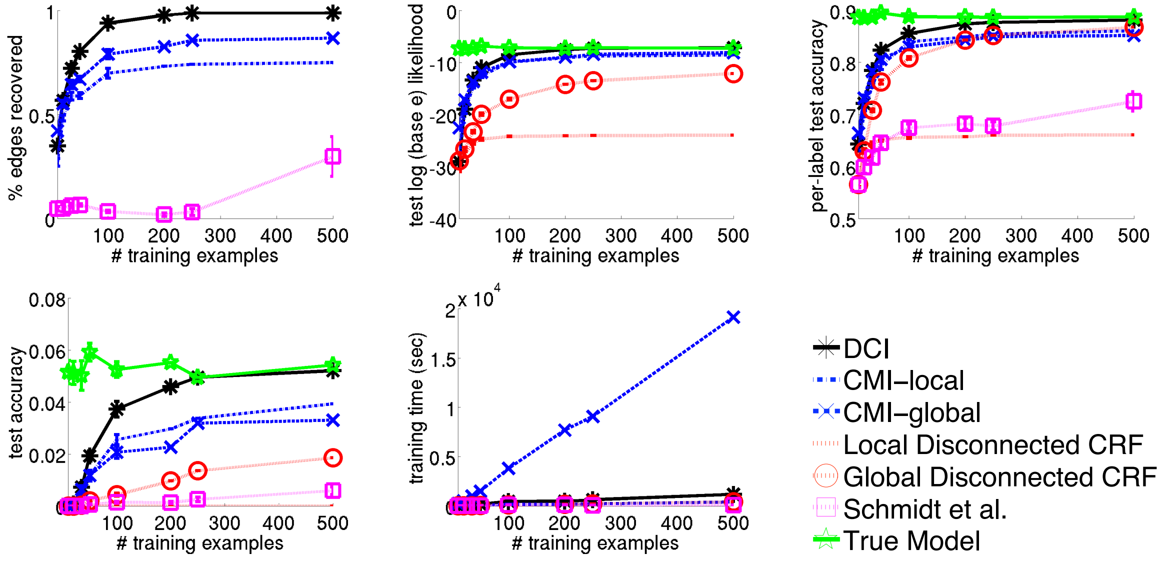


Figure 3.5: Synthetic data: Varying training set size. Tree CRFs $P(\mathcal{Y}|\mathcal{X})$ with associative factors. $|\mathcal{Y}| = |\mathcal{X}| = 40$. 1000 test examples. Averaged over 10 models/random samples; error bars (very small) show 2 standard errors.

CRF 2: We chose ≤ 20 inputs per Y_i; we added the same fixed factors. After structure learning, we parameterized edge factors to be independent of $\mathcal{X}$.

Palatucci et al. [2009] test *zero-shot learning*, which permits predictions about classes not seen during training. After predicting semantic features $\mathcal{Y}$ from images $\mathcal{X}$, they use hand-built "true" $\mathcal{Y}$ vectors to decode which object the test subject is thinking of. For testing, they use leave-2-out CV: Train on 58 objects; predict $\mathcal{Y}$ for 2 held-out objects i, j. Object i is classified correctly if its predicted $\widehat{\mathcal{Y}}^{(i)}$ is closer in ℓ_2 norm to its true $\mathcal{Y}^{(i)}$ than to the true $\mathcal{Y}^{(j)}$.

We used the same setup, scoring using their accuracy measure, squared error of predicted $\mathcal{Y}$, and log probability $\log P(\mathcal{Y}|\mathcal{X})$. Palatucci et al. [2009] use ridge regression $Y_i \sim \mathcal{X}$, $\forall i$, equivalent to a disconnected CRF with global inputs; we used this as a baseline.

Figure 3.8 compares disconnected and tree CRFs. The discrepancy between the 3 performance metrics is remarkable: Tree CRFs are best at predicting $\mathcal{Y}$ w.r.t. log likelihood and squared error (before decoding), but disconnected CRFs are best w.r.t. the accuracy metric (after decoding). This behavior could be caused by decoding via Euclidean distance and not accounting for the relative importance of each Y_i. We also tested decoding by predicting the more likely of the two held-out objects' $\mathcal{Y}$ vectors, but this performed worse with all learning methods. Learning this decoding $\mathcal{Y} \longrightarrow$ objects might avoid this problem and be a valuable addition to the zero-shot learning framework.

3.5 Discussion

Combining a maximum spanning tree algorithm with carefully chosen edge scores allows us to learn expressive models while avoiding the costliness of many structure learning methods. Despite our negative

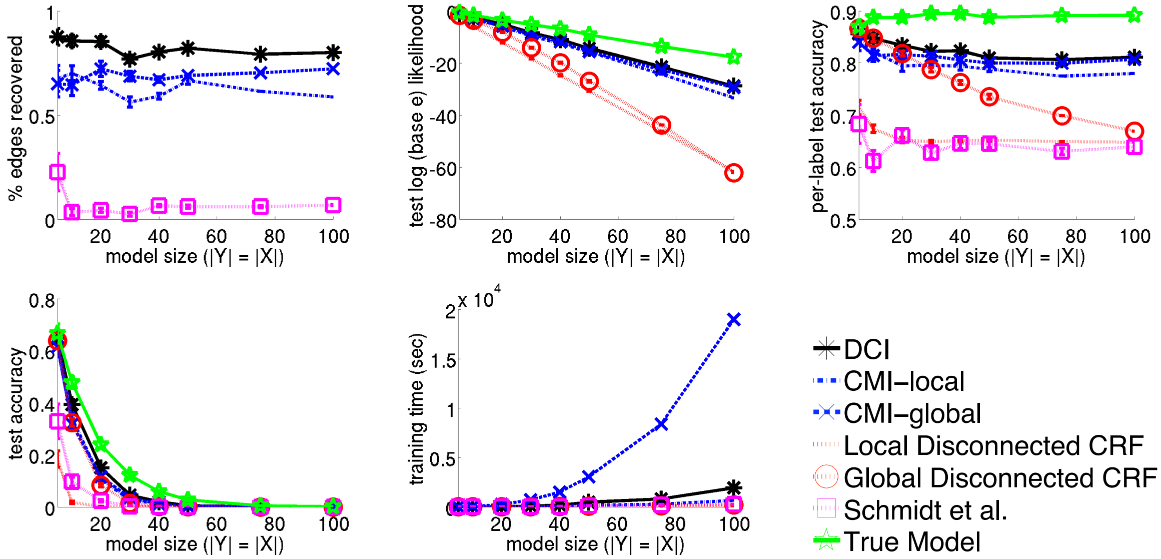


Figure 3.6: Synthetic data: Varying model size ($|\mathcal{Y}| = |\mathcal{X}|$). Tree CRFs $P(\mathcal{Y}|\mathcal{X})$ with associative factors. 50 training, 1000 test examples. Averaged over 10 models/random samples; error bars (very small) show 2 standard errors.

result for Local Linear Entropy Scores, local CMI and DCI scores can often recover the edges of tree CRFs.

Our structure learning work follows the theme of decomposition from our work on parameter learning, but further work will be needed to prove results about trade-offs between sample complexity, computation, and parallelization. We discuss ideas for improving our method in the next section.

3.6 Future Work

Although no Local Linear Entropy Score (LLES) (Sec. 3.2.2) suffices for all models, a fixed LLES might suffice for useful subclasses of models. Analysis demonstrating such subclasses of tree CRFs which are recoverable via fixed local scores would be worthwhile.

Several internal algorithmic improvements could be useful. In our tests, parameter learning was a bottleneck. Making use of our work on scalable parameter learning from Ch. 2 and parallel optimization from Ch. 4 would be very useful for future tests. (Our work on structure learning preceded these other works.) Also, we currently estimate entropies by estimating probability distributions and then computing the entropies. Better entropy estimates which avoid probability estimation would likely improve performance.

Our results on structure learning so far are mixed. While the synthetic results strongly support the use of LLESs for learning tree structures, the fMRI results are weaker. We propose improvements to our method which may help on real-world problems: choosing a data-specific LLES (Sec. 3.6.1) and learning evidence locality (Sec. 3.6.2). We also propose an extension to more general models (Sec. 3.6.3).

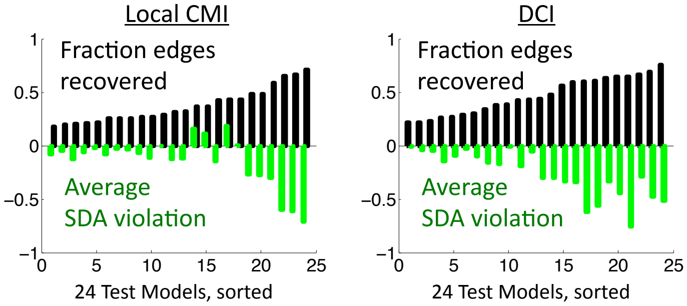


Figure 3.7: Score Decay Assumption violation vs. edge recovery on 24 models: ($|\mathcal{Y}| = 10, 15, 20$) × (with/out cross factors) × (2 associative factor types, 2 random). SDA violation averaged over consecutive triplets. For edge recovery, up=better; for SDA violation, down=better. 50 train exs. Tests averaged over 10 samples. Tractable $P(\mathcal{Y}, \mathcal{X})$.

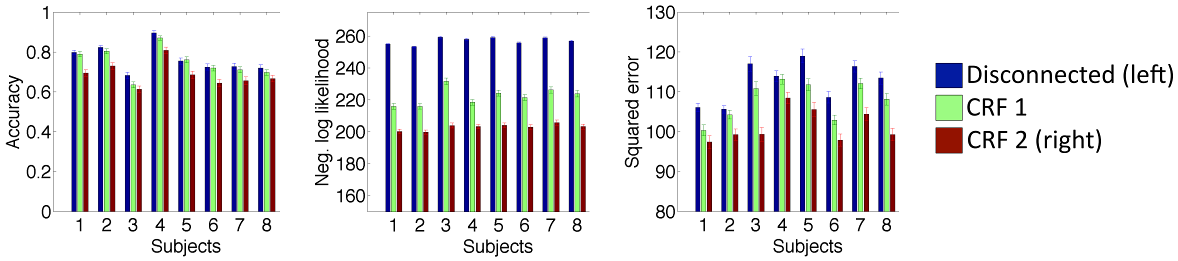


Figure 3.8: fMRI results. Error bars are 2 standard errors long. CRF structure and parameter learning with fixed regularization took only about 2-3 times as long as closed-form leave-one-out cross validation for ridge regression.

3.6.1 Learning Score Functions

Theorem 3.2.1 and Theorem 3.2.2 indicate that the ideal LLES (edge score) is problem-dependent. Selecting a score using training data might therefore produce better results. We sketch two approaches for selecting LLESs.

Approach 1: Our first approach is sketched in Alg. 3.2. We are given a set of trees, as well as a method for estimating the importance of each edge in a tree. This importance can depend on the entire tree and could be, e.g., the change in the log likelihood of the model from removing that edge. For each edge, we compute this importance, along with the local entropy terms used in LLESs. We then use regression to learn a LLES which predicts the edge importances from the entropy terms.

Many methods could be used to choose the set of trees given to Alg. 3.2:

- We could sample random trees. Alg. 3.2 would use equal weights for each tree.
- We could learn a mixture of trees via, e.g., the method for boosting mixtures of trees from Rosset and Segal [2002]. (Their work extends easily to CRFs.) Alg. 3.2 could make use of the mixture weights.
- We could iteratively construct a set of tree structures, alternating between re-learning a LLES and choosing a new tree structure.

In all of these methods, we would need to explore the very large space of tree structures.

Algorithm 3.2: Learn Local Linear Entropy Score (LLES) from Structures

```
Input: Dataset over $(Y, X)$; weighted set of tree CRFs $\{Q_k(\mathcal{Y}|\mathcal{X})\}_k$
foreach $k$ do
    foreach Edge $(Y_i, Y_j)$ in tree $Q_k$ do
        Create a sample $(A = a, B = b)$, with $a = Importance(Y_i, Y_j, Q_k)$ and $b =$ LLES
        entropies, weighted by $Q_k$'s given weight.
Regress $A \sim B$ using the generated samples to get a LLES $S$.
return LLES $S$
```

Approach 2: Our second approach is based on the observation that the LLES weights (i.e., the weight given to each local entropy in the LLES) are essentially auxiliary variables to aid in our optimization. We could try to pose the problem of learning an LLES and a structure as a joint optimization with a single objective. Our ideal objective would be the likelihood of our structure, which is in turn a function of the LLES. This second function (from LLES to structure) uses the max-weight spanning tree algorithm, and we would likely need to replace this non-convex part of our objective with a soft-max formulation. One possibility would be to use a mixture of trees, where the mixture weights are based on the LLES.

3.6.2 Learning Evidence Locality

Recall that evidence locality refers to the existence of small sets $X_D \in \mathcal{X}$ of variables participating in each factor. When evidence locality is known, utilizing the locality by parameterizing factors as $\psi(Y_C, X_D)$ with $|X_D| \ll |X|$ (instead of as $\psi(Y_C, X)$) offers clear computational and statistical benefits. When evidence locality is unknown, as in the fMRI application in Sec. 3.4.2, we would like to learn the locality from data. Ideally, we could learn the structure and evidence locality via a joint optimization.

Algorithm 3.3: Jointly Learning Structure and Local Evidence

```
Input: Dataset over $(Y, X)$
Choose initial local evidence $X_{D_i}$ for each $Y_i$.
while not converged do
    Learn structure over $\mathcal{Y}$, given fixed local evidence $\{X_{D_i}\}_i$.
    Choose new local evidence $\{X_{D_i}\}_i$, given fixed structure over $\mathcal{Y}$.
return Structure, Local evidence
```

We propose an initial algorithmic framework to test in Alg. 3.3. It is a block coordinate optimization method which iterates between choosing a structure given fixed local evidence and choosing local evidence given a fixed structure. The specific methods for learning the structure and for choosing local evidence may vary.

For the structure learning step, a natural first step would be to try the LLES + MST methods from Alg. 3.1. These, however, may cause joint learning to converge too quickly; i.e., the choice of a full tree, rather than, e.g., a forest, may force the optimization into a local optimum. If local optima cause problems, we propose to generalize our LLES + MST method to learning forests, rather than full trees; we could do so using the methods which Liu et al. [2010b] used for learning generative forests. Alg. 3.3's structure learning step would learn forests, slowly "cooling" the structure learning step in a way analogous to simulated annealing [Kirkpatrick et al., 1983] so that each iteration would add more edges to the forest. Intuitively, by choosing a smaller set of edges for our structure in intermediate iterations, we would commit less to a particular structure and could be less likely to get stuck in local optima.

For the evidence selection step, we propose two methods. The first, simpler method would select local evidence separately for each Y_i by regressing Y_i on its neighbors in $\mathcal{Y}$ (in the fixed structure) and on all of $\mathcal{X}$, with L1 regularization to encourage sparsity and so select small sets of local evidence. The results of Ravikumar et al. [2010] on recovering structures via L1-regularized regression provide theoretical motivation for this method. To speed up computation, we could take advantage of our work on parallel sparse regression in Ch. 4.

The second, more complex method will use max-likelihood CRF parameter learning with each factor in the fixed structure parameterized using all of $\mathcal{X}$ (i.e., global evidence); again, we will impose L1 regularization on parameters involved with $\mathcal{X}$ variables to encourage small sets of local evidence. This second method will permit more global choices of local evidence, and it may prove easier to analyze since it optimizes the very objective (log likelihood) which we are trying to optimize via the joint learning procedure, albeit with a different regularization term.

These evidence selection methods resemble the work by Schmidt et al. [2008] on structure learning via optimization of L1-regularized pseudolikelihood. Unlike them, we intend to learn low-treewidth structures and to use likelihood, not pseudolikelihood, in our optimization.

3.6.3 General Structures

While tree structures permit tractable inference, trees are not always expressive enough for certain applications. We propose combining our work with several existing works.

Treewidth-k Structures: Shahaf et al. [2009] proposed an efficient method for learning treewidth-k structures (with small k) which, like our methods, is based on edge weights. Our methods using Local Linear Entropy Scores for edge weights may be naturally combined with their algorithm for choosing treewidth-k structures.

L1-Regularized Pseudolikelihood and Composite Likelihood: Schmidt et al. [2008] and Ravikumar et al. [2010] propose learning PGM structure via L1-regularized pseudolikelihood (optimized jointly and disjointly, respectively). Our proposed extensions to parallel sparse regression in Ch. 4 could improve the scalability of their methods during structure learning. Given a learned structure, rather than using the parameters produced by pseudolikelihood, we could re-learn parameters via a more accurate method such as composite likelihood. This step could take advantage of our work on structured composite likelihood in Ch. 2.

Chapter 4

Parallel Regression

The CRF parameter and structure learning methods discussed in Ch. 2 and Ch. 3 largely reduce learning to solving many regression problems. Also, many of the methods for structure learning discussed as related work (Sec. 3.1) use L_1-regularized optimization, where the L_1 penalty biases learning towards sparser solutions (i.e., sparser structures). This chapter discusses work on parallelizing these regression problems to take advantage of the growing availability of parallel computing platforms.

We present `Shotgun`, a parallel stochastic coordinate descent method for L_1-regularized regression. The method is based on the existence of data-specific locality in the problem which limits the effect of one coordinate update on other coordinate updates. This locality allows us to greatly increase parallelism while keeping total computation essentially unchanged (thus achieving near-linear speedups). We discuss extensions to the distributed setting in Sec. 4.7, and our overarching future goals (Ch. 6) are heavily based on ideas from this initial work.

Our main contributions in this chapter may be summarized as follows:

- `Shotgun`, a simple algorithm for parallel coordinate descent
- Convergence analysis indicating near-linear speedups, up to a problem-dependent limit
  - Discussion of generalized analysis and learning problems
- Practical implementation of `Shotgun` and extensive experiments which show:
  - Our theory (convergence bounds) accurately predict empirical performance.
  - `Shotgun` is one of the most scalable sparse regression methods for multicore.
  - `Shotgun` achieves significant speedups.

Our work on `Shotgun` was initially presented in Bradley et al. [2011].

4.1 Introduction

Many applications in machine learning and statistics use L_1-regularized models such as the Lasso [Tibshirani, 1996] and sparse logistic regression [Ng, 2004]. L_1 regularization biases learning towards sparse solutions, and it is especially useful for *high-dimensional* problems with large numbers of features. For example, in logistic regression, it allows sample complexity to scale logarithmically w.r.t. the number of irrelevant features [Ng, 2004]. Sparse solutions have also been a focus of research in the domains of compressed sensing, digital image processing, and inverse problems in various branches of science. (See generally Bruckstein et al. [2009].)

Much effort has been put into developing optimization algorithms for L_1 models. These algorithms range from iterative thresholding/shrinkage methods [Wen et al., 2010, Wright et al., 2009] and stochastic gradient [Shalev-Schwartz and Tewari, 2009] to more computationally intensive interior point methods [Kim et al., 2007].

Coordinate descent, which we call `Shooting` after Fu [1998], is a simple but very effective algorithm which updates one coordinate per iteration, performing minimization w.r.t. that coordinate while holding all other coordinates fixed. As we discuss in Sec. 4.2, theory [Shalev-Schwartz and Tewari, 2009] and extensive empirical results [Yuan et al., 2010] have shown that variants of `Shooting` are particularly competitive on high-dimensional data. In addition, coordinate descent does not require careful tuning of parameters, unlike methods such as stochastic gradient which require selecting learning rates.

In this work, we are interested in developing faster algorithms for solving large-scale L_1-regularized problems, especially when the training samples have sparse values. The need for scalable optimization is growing as more applications use high-dimensional data, but processor core speeds have stopped increasing in recent years. Instead, computers come with more cores, and the new challenge is utilizing them efficiently. For this, we turn to parallel optimization algorithms.

We began our research with an extensive experimental evaluation of current algorithms for minimizing L_1-regularized squared error and logistic loss. In the process, we found that `Shooting` was competitive on almost all types of problems and outperformed other algorithms on large and sparse datasets. The natural question was whether we could improve `Shooting`—or another algorithm—via parallelization.

Parallelizing internal operations: Most other algorithms, such as interior point methods and iterative shrinkage and thresholding algorithms, were already benefiting from parallel matrix-vector operations in their underlying linear algebra libraries. Parallelizing the internal operations of `Shooting` proved to be inefficient because each update involves only a handful of numerical operations. Particularly with sparse input, the overhead required for parallel computation outweighed the benefits.

Parallelizing over samples: We also studied recent work which analyzes parallel stochastic gradient descent for multicore [Langford et al., 2009b] and distributed settings [Mann et al., 2009, Zinkevich et al., 2010]. These methods parallelize over samples. In applications using L_1 regularization, though, there are often many more features than samples, so parallelizing over samples may be of limited utility.

Parallelizing over features: Thus, we were left with a final natural direction for parallelizing `Shooting`: minimizing multiple coordinates in parallel. At first sight, this seems unreasonable because of cross-feature correlations. However, after a series of experiments, we were surprised to discover that parallel `Shooting` with up to 16 processor cores converged faster than sequential `Shooting` on most problems, and that the few failure cases exhibited rapid divergence. In our subsequent analysis, we were able to explain this behavior theoretically and confirm our theory's predictions in experiments.

In Sec. 4.3, we describe `Shotgun` [Bradley et al., 2011], a simple multicore version of `Shooting` which updates `P` coordinates in parallel. We prove strong convergence bounds for `Shotgun` predicting speedups over `Shooting` which are near-linear in `P`, up to a problem-dependent optimum `P`*. Moreover, our theory provides an estimate for this ideal `P`* which may be easily computed from the data.

In Sec. 4.4, we present a brief survey of related work on optimization algorithms for problems with L_1 regularization. Since our initial work [Bradley et al., 2011], there has been a significant amount of new research in the field, including new distributed algorithms for solving convex problems. In our experiments, these new methods are promising for solving very large problems that cannot fit into the memory of a single computer, but they are not as fast as `Shotgun` in the multicore setting.

In Sec. 4.5, we compare multicore `Shotgun` with five state-of-the-art algorithms on 35 real and synthetic datasets. The results show that in large problems `Shotgun` outperforms the other algorithms. The experiments also validate our theoretical predictions by showing that `Shotgun` requires only about 1/`P` as many iterations as `Shooting`. We measure the parallel speedup in running time and analyze the limitations imposed by current multicore hardware.

This paper improves upon our initial results [Bradley et al., 2011] with corrections and extensions to theory and experiments, as well as an extensive discussion comparing our method with related work.

4.2 L_1-Regularized Loss Minimization

We consider classification and regression problems in which we wish to use a d-dimensional *feature vector* $\mathbf{a}_i \in \mathbb{R}^d$ to predict an outcome $y_i \in \mathcal{Y}$. Learning such a prediction function commonly involves minimizing a regularized loss over training samples, leading to optimization problems of the form

$$\min_{\mathbf{x}\in\mathbb{R}^d} F(\mathbf{x}), \text{ where } F(\mathbf{x}) = \sum_{i=1}^{n} L(\mathbf{a}_i^T\mathbf{x},\, y_i) + \lambda\|\mathbf{x}\|_1 \,. \tag{4.1}$$

Above, $L(\cdot)$ is the loss, which we assume to be non-negative and convex. The function is parameterized by $\mathbf{x} \in \mathbb{R}^d$, an *unknown* vector of *weights* for features. We have n training *samples*, each of which is specified by the features $\mathbf{a}_i$ and outcome $y_i \in \mathcal{Y}$. We weight the L_1 regularization term with the parameter $\lambda \geq 0$. $F(\mathbf{x})$ is called the *objective function*.

We let $\mathbf{A} \in \mathbb{R}^{n\times d}$ be the design matrix, whose i^{th} row is $\mathbf{a}_i$. We assume w.l.o.g. that columns of $\mathbf{A}$ are normalized s.t. $diag(\mathbf{A}^T\mathbf{A}) = \mathbf{1}$.[1] We denote $\mathbf{y} \in \mathcal{Y}^n$ as the vector of outcomes for all samples.

One instance of Eq. (4.1) is the Lasso [Tibshirani, 1996], which uses the squared error with $\mathcal{Y} \equiv \mathbb{R}$:

$$\min_{\mathbf{x}\in\mathbb{R}^d} \tfrac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1 \,. \tag{4.2}$$

Another instance of Eq. (4.1) is sparse logistic regression [Ng, 2004], which uses the logistic loss with $\mathcal{Y} \equiv \{-1, +1\}$:

$$\min_{\mathbf{x}\in\mathbb{R}^d} \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i\mathbf{a}_i^T\mathbf{x}\right)\right) + \lambda\|\mathbf{x}\|_1 \,. \tag{4.3}$$

For analysis, we follow Shalev-Schwartz and Tewari [2009] and transform Eq. (4.1) into an equivalent problem with a twice-differentiable regularizer. We use *duplicated features* $\hat{\mathbf{a}}_i = [\mathbf{a}_i; -\mathbf{a}_i] \in \mathbb{R}^{2d}$ and let

[1] Normalizing $\mathbf{A}$ does not change the objective if a separate, normalized λ_j is used for each x_j. If column j of $\mathbf{A}$ is scaled by $1/z$, then we scale $\lambda_j = \lambda/z$; scaling x_j by z will then leave the objective unchanged. We use this trick in our experiments.

Algorithm 4.1: `Shooting`: Sequential `SCD`

```
Input: Data $\mathbf{A} \in \mathbb{R}^{n \times d}$ , $\mathbf{y} \in \mathbb{R}^n$; scalar $\lambda \geq 0$
Set $\mathbf{x} = \mathbf{0} \in \mathbb{R}_+^{2d}$.
while not converged do
    Choose $j \in \{1, \ldots, 2d\}$ uniformly at random.
    Set $\delta x_j \longleftarrow \max\{-x_j, -(\nabla F(\mathbf{x}))_j / \beta\}$.
    Update $x_j \longleftarrow x_j + \delta x_j$.
```

$\hat{\mathbf{x}} \in \mathbb{R}_+^{2d}$, where $\mathbb{R}_+$ denotes the non-negative real numbers. We solve

$$\min_{\hat{\mathbf{x}} \in \mathbb{R}_+^{2d}} \sum_{i=1}^{n} L(\hat{\mathbf{a}}_i^T \hat{\mathbf{x}}, \, y_i) + \lambda \sum_{j=1}^{2d} \hat{x}_j \, . \tag{4.4}$$

If $\hat{\mathbf{x}} \in \mathbb{R}_+^{2d}$ minimizes Eq. (4.4), then $\mathbf{x} : \, x_i \doteq \hat{x}_{d+i} - \hat{x}_i$ minimizes Eq. (4.1). We prove this equivalence in more detail in Sec. B.1.2. Though our analysis uses duplicate features, they are not needed for an implementation.

4.2.1 Sequential Stochastic Coordinate Descent (Sequential `SCD`)

Shalev-Schwartz and Tewari [2009], Shalev-Shwartz and Tewari [2011] analyze `Stochastic Coordinate Descent (SCD)`, a stochastic version of `Shooting` for solving Eq. (4.1). `SCD`, detailed in Alg. 4.1, randomly chooses one weight x_j to update per iteration. `SCD` computes the update $x_j \leftarrow x_j + \delta x_j$ via

$$\delta x_j = \max\{-x_j, \, -(\nabla F(\mathbf{x}))_j / \beta\} \, , \tag{4.5}$$

where $\beta > 0$ is a loss-dependent constant. β is an upper bound on the curvature of the loss, formalized by the following assumption which places a uniform upper bound on the change in the loss $F(\mathbf{x})$ from updating a single weight.

Assumption 4.2.1. *Let $F(\mathbf{x}) : \mathbb{R}_+^{2d} \longrightarrow \mathbb{R}$ be a convex function. Let $\mathbf{e}_j$ be a unit vector with 1 in its j^{th} entry. Assume there exists $\beta > 0$ s.t., for all $\mathbf{x}$ and single-weight updates δx_j, we have:*

$$F(\mathbf{x} + (\delta x_j)\mathbf{e}_j) \leq F(\mathbf{x}) + \delta x_j (\nabla F(\mathbf{x}))_j + \tfrac{\beta}{2}(\delta x_j)^2 \, .$$

For the losses in Eq. (4.2) and Eq. (4.3), Shalev-Schwartz and Tewari [2009] show that Taylor expansions give

$$\beta = 1 \text{ (squared error) and } \beta = \tfrac{1}{4} \text{ (logistic loss).} \tag{4.6}$$

We provide a proof of Eq. (4.6) in Sec. B.1.1. With these values for β, the closed-form update in Eq. (4.5) is an exact minimization for the Lasso (Eq. (4.2)) and is guaranteed to decrease the objective for logistic regression (Eq. (4.3)).

Using Assumption 4.2.1, Shalev-Schwartz and Tewari [2009] prove the following convergence bound for `SCD`. (We also provide a proof in Sec. B.1.3.)

Theorem 4.2.1. *[Shalev-Schwartz and Tewari, 2009] Let $\mathbf{x}^*$ minimize Eq. (4.4) and $\mathbf{x}^{(T)}$ be the output of Alg. 4.1 after T iterations. If $F(\mathbf{x})$ satisfies Assumption 4.2.1, then*

$$\mathbf{E}\left[F(\mathbf{x}^{(T)})\right] - F(\mathbf{x}^*) \leq \frac{d(\beta\|\mathbf{x}^*\|_2^2 + 2F(\mathbf{x}^{(0)}))}{T+1}\,, \tag{4.7}$$

where $\mathbf{E}[\cdot]$ is w.r.t. the random choices of weights j.

Theorem 4.2.1 shows that the distance of the expected objective on iteration T from the optimum decreases as $1/T$. The bound increases with $\|\mathbf{x}^*\|_2^2$, which accounts for the distance between the initial weights $\mathbf{x}^{(0)} = \mathbf{0}$ and the optimum $\mathbf{x}^*$, and increases with $F(\mathbf{x}^{(0)})$, the initial objective value.

4.2.2 Scalability of SCD

Multiple works have argued that SCD and other variants of coordinate descent are among the most scalable methods for solving sparse regression problems of the form in Eq. (4.1). Yuan et al. [2010] present a large-scale empirical comparison of algorithms for sparse logistic regression, including coordinate descent, block coordinate descent, trust region Newton methods, and interior point methods. Their results consistently highlight coordinate descent as one of the fastest methods.

As Shalev-Schwartz and Tewari [2009] argue, Theorem 4.2.1 indicates that SCD scales well, relative to other methods, as the dimensionality d of the data increases. Nesterov [2010] also provides theoretical results indicating that SCD scales well w.r.t. d, and discusses classes of objectives for which coordinate descent can outperform full-gradient updates. We compare SCD with other methods in more detail in Sec. 4.4.

We emphasize that our work parallelizes one of the fastest sequential methods for sparse regression. As our empirical results show, slower methods may be simpler to parallelize (e.g., by parallelizing operations on large matrices and vectors), but those slow methods are often outperformed by even sequential SCD.

4.3 Parallel Coordinate Descent

As the dimensionality d or sample size n increase, even fast sequential algorithms become expensive. To scale to larger problems, we turn to parallel computation, using the multicore setting. In this section, we present our main theoretical contribution: we prove that coordinate descent can be parallelized, giving strong convergence bounds.

We parallelize Shooting (SCD) and call our algorithm Shotgun (Alg. 4.2). Shotgun initially chooses P, the number of weights to update in parallel. On each iteration, it chooses a subset of P weights from $\{1, \ldots, 2d\}$ uniformly at random from the possible combinations; these form a set $\mathcal{P}_t$. It updates each $x_i : i \in \mathcal{P}_t$, in parallel using the same update as Shooting Eq. (4.5). We define $\Delta\mathbf{x}$ as the collective update to $\mathbf{x}$; i.e., $(\Delta\mathbf{x})_i = \delta x_i$ if $i \in \mathcal{P}_t$, and $(\Delta\mathbf{x})_i = 0$ otherwise.

Intuitively, parallel updates might increase the risk of divergence. In Fig. 4.1, the left subplot shows how parallel updates can speed up convergence when features are uncorrelated; the right subplot shows how parallel updates of correlated features can increase the objective. We could avoid such divergence by imposing a step size; e.g., a step size of $\frac{1}{\mathtt{P}}$ ensures convergence since F is convex in $\mathbf{x}$. However,

Algorithm 4.2: `Shotgun`: Parallel `SCD`

```
Input: Data $\mathbf{A} \in \mathbb{R}^{n \times d}$, $\mathbf{y} \in \mathbb{R}^n$; scalar $\lambda \geq 0$
Choose number of parallel updates P $\geq 1$.
Set $\mathbf{x} = \mathbf{0} \in \mathbb{R}^{2d}_{+}$.
while not converged do
    Choose random subset of P weights in $\{1, \ldots, 2d\}$.
    In parallel on P processors do
        Get assigned weight $j$.
        Set $\delta x_j \longleftarrow \max\{-x_j, -(\nabla F(\mathbf{x}))_j/\beta\}$.
        Update $x_j \longleftarrow x_j + \delta x_j$.
```

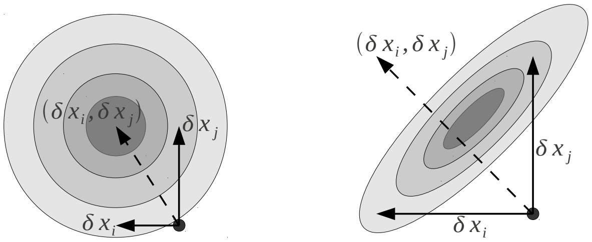


Figure 4.1: **Intuition for parallel coordinate descent.** Contour plots of two objectives, with darker meaning better. Each plot shows updates δ_i, δ_j which minimize coordinates independently, plus the sum of the updates when executed in parallel. Left: The features are uncorrelated, so parallel updates do not conflict and speed convergence. Right: The features are correlated, so parallel updates conflict and possibly cause divergence.

our experiments showed that approach to be impractical, for it results in very small steps and long runtimes.

We formalize this intuition for the Lasso in Theorem 4.3.1. We can separate a sequential progress term (summing the improvement from separate updates) from a term measuring interference between parallel updates. If $\mathbf{A}^T\mathbf{A}$ were normalized and centered to be a covariance matrix, the elements in the interference term's sum would be non-zero only for correlated variables, matching our intuition from Fig. 4.1. Harmful interference could occur when, e.g., $\delta x_i, \delta x_j > 0$ and features i, j were positively correlated.

Theorem 4.3.1. *Fix* $\mathbf{x}$. *If* $\Delta\mathbf{x}$ *is the collective update to* $\mathbf{x}$ *in one iteration of Alg. 4.2 for the Lasso, then*

$$F(\mathbf{x} + \Delta\mathbf{x}) - F(\mathbf{x}) \quad \leq \quad \underbrace{-\tfrac{1}{2} \sum_{i \in \mathcal{P}_t} (\delta x_i)^2}_{\textit{sequential progress}} + \underbrace{\tfrac{1}{2} \sum_{i,j \in \mathcal{P}_t} (\mathbf{A}^T\mathbf{A})_{i,j} \delta x_i \delta x_j}_{\textit{interference}} . \tag{4.8}$$

We prove Theorem 4.3.1 in Sec. B.1.4. In the next section, we show that this intuition holds for the more general optimization problem in Eq. (4.1).

4.3.1 `Shotgun` Convergence Analysis

In this section, we present our convergence result for `Shotgun`. The result provides a problem-specific measure of the potential for parallelization: the spectral radius ρ of $\mathbf{A}^T\mathbf{A}$ (i.e., the maximal absolute eigenvalue of $\mathbf{A}^T\mathbf{A}$). Moreover, this measure is prescriptive: ρ may be estimated via methods like power iteration[2] [Gilbert, 1988], and it provides a plug-in estimate of the ideal number of parallel updates.

We begin by generalizing Assumption 4.2.1 to our parallel setting. The assumption still holds for Lasso and logistic regression, with the same scalars β as in Eq. (4.6).

Assumption 4.3.1. *Let $F(\mathbf{x}) : \mathbb{R}_+^{2d} \longrightarrow \mathbb{R}$ be a convex function. Assume that there exists $\beta > 0$ such that, for all $\mathbf{x}$ and parallel updates $\Delta\mathbf{x}$, we have*

$$F(\mathbf{x} + \Delta\mathbf{x}) \leq F(\mathbf{x}) + \Delta\mathbf{x}^T \nabla F(\mathbf{x}) + \tfrac{\beta}{2}\Delta\mathbf{x}^T\mathbf{A}^T\mathbf{A}\Delta\mathbf{x}\,.$$

Similarly to Theorem 4.3.1, the above assumption bounds the change in our objective $F(\cdot)$ when we update our weights $\mathbf{x}$ with $\Delta\mathbf{x}$. Our analysis will balance the progress measured by the first-order term $\Delta\mathbf{x}^T\nabla F(\mathbf{x})$ against harmful interference bounded by the second-order term. We now state our main result, generalizing the convergence bound in Theorem 4.2.1 to the `Shotgun` algorithm.

Theorem 4.3.2. *Let $\mathbf{x}^*$ minimize Eq. (4.4), and let $\mathbf{x}^{(T)}$ be the output of Alg. 4.2 after T iterations with $\mathtt{P}$ parallel updates per iteration. Let ρ be the spectral radius of $\mathbf{A}^T\mathbf{A}$. If $F(\mathbf{x})$ satisfies Assumption 4.3.1 and $\mathtt{P}$ is chosen s.t. $\epsilon \doteq \frac{(\mathtt{P}-1)(\rho-1)}{2d-1} < 1$, then*

$$\mathbf{E}\left[F(\mathbf{x}^{(T)})\right] - F(\mathbf{x}^*) \leq \frac{d\left(\beta\|\mathbf{x}^*\|_2^2 + \frac{2}{1-\epsilon}F(\mathbf{x}^{(0)})\right)}{(T+1)\mathtt{P}}\,,$$

where $\mathbf{E}[\cdot]$ is w.r.t. the random choices of weights to update. Choosing a near-optimal $\mathtt{P}^* \approx \frac{d}{\rho}$ gives

$$\mathbf{E}\left[F(\mathbf{x}^{(T)})\right] - F(\mathbf{x}^*) \lesssim \frac{\rho\left(\beta\|\mathbf{x}^*\|_2^2 + 4F(\mathbf{x}^{(0)})\right)}{T+1}\,.$$

Without duplicated features, Theorem 4.3.2 predicts that we can do up to $\mathtt{P} \leq \frac{d}{2\rho}$ parallel updates and achieve speedups almost linear in $\mathtt{P}$. For an ideal problem with uncorrelated features, $\rho = 1$, so we could do up to $\mathtt{P}^* = d$ parallel updates. For a pathological problem with exactly correlated features, $\rho = d$, so our theorem tells us that we could not do parallel updates. With $\mathtt{P} = 1$, we recover the result for `Shooting` in Theorem 4.2.1.

The choice $\mathtt{P}^*$ is near-optimal since it is within a factor 2 of the maximum $\mathtt{P}$ s.t. $\epsilon < 1$ and since it sets $\epsilon \approx \frac{1}{2}$ in the bound (so ϵ is a small constant). When we have unlimited parallelism and use this near-optimal $\mathtt{P}^*$, the convergence bound ceases to depend on the dimensionality d and instead depends on $\rho \in [1, d]$.

In Sec. B.1.7, we discuss choosing weights independently to form a multiset. We can produce a convergence bound analogous to that in Theorem 4.3.2, but it has larger constants and permits fewer parallel updates.

To prove Theorem 4.3.2, we first bound the harmful impact of interference between parallel updates.

[2] For our datasets, power iteration gave reasonable estimates of ρ within a small fraction of the total `Shotgun` runtime.

Lemma 4.3.3. *Fix* $\mathbf{x}$*. Let* $\epsilon = \frac{(\mathtt{P}-1)(\rho-1)}{2d-1}$*, where* $\mathtt{P}$ *is chosen s.t.* $\epsilon < 1$*. Let* $\mathcal{P}_t$ *be the set of coordinates updated, and let* $\Delta\mathbf{x}$ *be the collective update to coordinates* $\mathcal{P}_t$ *of* $\mathbf{x}$ *in one iteration of Alg. 4.2. Let* δx_j *be the update to coordinate* j *given by Eq. (4.5). Under the assumptions and definitions from Theorem 4.3.2,*

$$\mathbf{E}_{\mathcal{P}_t}\left[F(\mathbf{x}+\Delta\mathbf{x}) - F(\mathbf{x})\right] \leq \mathtt{P}\cdot\mathbf{E}_j\left[\delta x_j(\nabla F(\mathbf{x}))_j + \tfrac{\beta}{2}(1+\epsilon)(\delta x_j)^2\right], \tag{4.9}$$

where $\mathbf{E}_{\mathcal{P}_t}$ *is w.r.t. the random choice of* $\mathcal{P}_t$ *and* $\mathbf{E}_j$ *is w.r.t. choosing* $j \in \{1,\ldots,2d\}$ *uniformly at random.*

The above lemma shows that we make progress as long as ϵ is small enough. In the expectation on the right-hand side of Eq. (4.9), the first term $\delta x_j(\nabla F(\mathbf{x}))_j$ will be negative (good), while $(\delta x_j)^2$ will be positive (bad). If ϵ is small enough s.t. the sum of terms is negative, then we will make progress in expectation.

We prove Lemma 4.3.3 and Theorem 4.3.2 in Sec. B.1.6 and Sec. B.1.5, respectively. The proof for Lemma 4.3.3 introduces the spectral radius of $\mathbf{A}^T\mathbf{A}$ to bound the harmful interference from the second-order term in Assumption 4.3.1. The proof for Theorem 4.3.2 is similar to that for Theorem 4.2.1, but it replaces Assumption 4.2.1 with Lemma 4.3.3 and uses a different potential function.

4.3.2 Theory vs. Empirical Performance

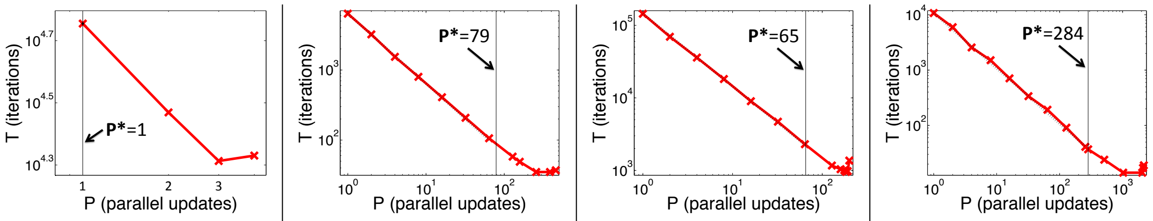
We briefly compare the predictions of Theorem 4.3.2 about the number of parallel updates P with the actual empirical performance for Lasso. We exactly simulated Shotgun as in Alg. 4.2 to eliminate effects from the practical implementation choices made in Sec. 4.5. We tested two single-pixel camera datasets from Duarte et al. [2008] and two compressed sensing datasets from the Sparco testbed [van den Berg et al., 2009], with a wide range of spectral radii ρ. We estimated $\mathbf{E}_{P_t}\left[F(\mathbf{x}^{(T)})\right]$ by averaging 10 runs of Shotgun. For Ball64_singlepixcam, we used $\lambda = 0.5$ to get $\mathbf{x}^*$ with about 68% non-zeros; for Mug32_singlepixcam, we used $\lambda = 0.05$ to get about 45% non-zeros; for SparcoProblem5, we used $\lambda = 0.5$ to get about 29% non-zeros; and for SparcoProblem7, we used $\lambda = 0.5$ to get about 3% non-zeros.

Fig. 4.2 plots P versus the iterations T required for $\mathbf{E}_{P_t}\left[F(\mathbf{x}^{(T)})\right]$ to come within 0.5% of the optimum $F(\mathbf{x}^*)$. Theorem 4.3.2 predicts that T should decrease as $\frac{1}{\mathtt{P}}$ as long as $\mathtt{P} \leq \mathtt{P}^* \approx \frac{d}{2\rho}$. The empirical behavior follows this theory: using the predicted P* gives almost optimal speedups, and speedups are almost linear in P. As P exceeds P*, Shotgun soon diverges.

Fig. 4.2 confirms Theorem 4.3.2's result: Shooting, a seemingly sequential algorithm, can be parallelized and achieve near-linear speedups, and the spectral radius of $\mathbf{A}^T\mathbf{A}$ succinctly captures the potential for parallelism in a problem. To our knowledge, our convergence results are the first for parallel coordinate descent for L_1-regularized losses, and they apply to any convex loss satisfying Assumption 4.3.1.

4.3.3 Relaxing the Spectral Conditions on $\mathbf{A}^T\mathbf{A}$

Our required bound on the maximum eigenvalue of $\mathbf{A}^T\mathbf{A}$ is related to conditions required by many analyses of sparse signal recovery, such as the Restricted Isometry Property (RIP) of Candes and Tao [2005] and the restricted eigenvalue conditions of Meinshausen and Yu [2009]. These other conditions generally place bounds on the eigenvalues of submatrices of $\mathbf{A}^T\mathbf{A}$, rather than the eigenvalues of the entire matrix



Ball64_singlepixcam
$d = 4096$, $\rho = 2047.8$

Mug32_singlepixcam
$d = 1024$, $\rho = 6.4967$

SparcoProblem5
$d = 2048$, $\rho = 15.673$

SparcoProblem7
$d = 2560$, $\rho = 4.5113$

Figure 4.2: **Theory vs. empirical performance for Shotgun's P.** We compare the predictions of Theorem 4.3.2 with results for Lasso on four datasets. The Y-axis has iterations T until $\mathbf{E}_{P_t}[F(\mathbf{x}^{(T)})]$ came within 0.5% of $F(\mathbf{x}^*)$. Thick red lines plot T against increasing P (until too large P caused divergence). Thin vertical lines mark the near-optimal $\mathtt{P}^*$. Dotted diagonal lines show optimal (linear) speedups (and are mostly hidden by the actual results).

$\mathbf{A}^T\mathbf{A}$. For example, Candes and Tao [2005] say that $\mathbf{A}$ satisfies the RIP of order k with constant $\gamma_k < 1$ if γ_k is the smallest number s.t.

$$(1 - \gamma_k)\|\mathbf{x}\|_2^2 \leq \|\mathbf{A}\mathbf{x}\|_2^2 \leq (1 + \gamma_k)\|\mathbf{x}\|_2^2\,, \quad \forall \mathbf{x} \,:\, \|\mathbf{x}\|_0 \leq k\,. \tag{4.10}$$

We show here how our condition on the spectral radius of $\mathbf{A}^T\mathbf{A}$ may be similarly relaxed to submatrices of $\mathbf{A}^T\mathbf{A}$. The spectral radius ρ which we use in Theorem 4.3.2 may be expressed as the smallest number s.t. $\|\mathbf{A}\mathbf{x}\|_2^2 \leq \rho\|\mathbf{x}\|_2^2$, $\forall \mathbf{x}$. One may see the similarity with the right-hand inequality in Eq. (4.10), which bounds the spectral radius of k-by-k submatrices of $\mathbf{A}^T\mathbf{A}$. In fact, our proofs only use this inequality with P-sparse update vectors $\Delta\mathbf{x}$. Therefore, we can relax our required condition by replacing ρ with $\rho_{\mathtt{P}}$ in Theorem 4.3.2, where $\rho_{\mathtt{P}}$ is defined as the smallest number s.t.:

$$\|\mathbf{A}\mathbf{x}\|_2^2 \leq \rho_{\mathtt{P}}\|\mathbf{x}\|_2^2\,, \quad \forall \mathbf{x} \,:\, \|\mathbf{x}\|_0 \leq \mathtt{P}\,. \tag{4.11}$$

This condition is looser since $\rho_{\mathtt{P}} \leq \rho$. The difference between $\rho_{\mathtt{P}}$ and ρ is especially large when $d \gg n$ and P is relatively small ($p < n$); see, e.g., Candes and Tao [2006] for a discussion of such matrices in terms of the RIP.

4.3.4 Beyond L_1

Thus far, we have discussed Shotgun in terms of the Lasso and sparse logistic regression, but our results actually generalize to a wider class of objectives, including other smooth losses and L_2 regularization. The main requirement on the objective in Theorem 4.3.2 is Assumption 4.3.1. This assumption may be generalized by leaving the bound on the Hessian term in a more general form:

Assumption 4.3.2. *Let $F(\mathbf{x}) : \mathbb{R}_+^{2d} \longrightarrow \mathbb{R}$ be a convex function. Assume that there exists a scalar $\beta > 0$ and a positive semidefinite matrix $\mathbf{M}$ with an all-ones diagonal such that, for all $\mathbf{x}$ and parallel updates $\Delta\mathbf{x}$, we have*

$$F(\mathbf{x} + \Delta\mathbf{x}) \leq F(\mathbf{x}) + \Delta\mathbf{x}^T\nabla F(\mathbf{x}) + \tfrac{\beta}{2}\Delta\mathbf{x}^T\mathbf{M}\Delta\mathbf{x}\,.$$

After this generalization, Lemma 4.3.3 and Theorem 4.3.2 remain almost identical, with the only change being that the matrix $\mathbf{A}^T\mathbf{A}$ is replaced with its generalization $\mathbf{M}$. The proofs likewise remain the same.

(We normalize features in order to make $\mathbf{M}$ have an all-ones diagonal; after this normalization, the value of β may be identified. The update rule in Eq. (4.5) remains the same.)

In this paper, we focus on L_1 regularization since both sparse regression and our method (coordinate descent) are arguably most useful for high-dimensional settings. We focus on the squared error and logistic losses for their importance in the field. Also, these two losses result in the simple form $\mathbf{M} = \mathbf{A}^T\mathbf{A}$ and give values of β which do not depend on the data.

4.4 Related Work

In this section, we compare our `SCD`-based approach with alternative methods for L_1-regularized loss minimization. The literature on sparse regression, compressed sensing, and similar applications is vast, so we concentrate on recent work and try to touch on the main classes of algorithms. Each subsection discusses a major design choice which affects algorithms' convergence rates and potential for parallelization. We summarize in Tab. 4.1 and empirically evaluate many of the algorithms in Sec. 4.5.

We refer the interested reader to several other works with useful literature reviews. Yang et al. [2010] discuss some of the algorithms we include below, but w.r.t. a different Lasso-like problem formulation: $\min_{\mathbf{x}} \|\mathbf{x}\|_1 \, s.t. \, \mathbf{A}\mathbf{x} = \mathbf{y}$. They also provide some useful empirical comparisons. Yuan et al. [2010] empirically compare a large number of algorithms for sparse logistic regression. Figueiredo et al. [2008] discuss several problem formulations related to the Lasso penalty formulation in Eq. (4.2), as well as the applications and algorithms associated with each formulation.

4.4.1 Coordinate vs. Full Gradient Methods

Most algorithms in the literature are *full gradient methods*, which update all weights $\mathbf{x}$ at once.[3] *Coordinate Descent (CD) methods* update elements of $\mathbf{x}$ individually. These methods are very cheap per iteration when the gradient computation decomposes across coordinates, as it does for the squared error and logistic loss with L_1 regularization.

For coordinate descent, a major design choice is the order in which coordinates are updated. `Stochastic CD (SCD)` [Shalev-Schwartz and Tewari, 2009] chooses coordinates randomly, and it is generally the most efficient and easiest to analyze. `Cyclic CD`, which cycles through coordinates, performs similarly to `SCD` in practice, but its current convergence analysis requires an isotonicity assumption which does not hold in general [Saha and Tewari, 2010]. (For Lasso, the isotonicity assumption is equivalent to assuming that $(\mathbf{A}^T\mathbf{A})_{ij} \leq 0, \, \forall i \neq j$.)

`Greedy CD` updates the coordinate whose value changes most when optimized. Though `Greedy CD` requires fewer iterations in general than `SCD` and `Cyclic CD`, each iteration can be d times more expensive since the full gradient must be calculated. Dhillon et al. [2011] use approximate nearest neighbor search to reduce the expense of each iteration, making `Greedy CD` sublinear in the number of non-zero entries in $\mathbf{A}$ after a linear-time initialization. Their experiments show this improvement makes `Greedy CD` perform about as well as `Cyclic CD` on large problems. Homotopy methods, which include some

[3] We write "gradient" instead of "subgradient" since many terms in the literature are written with "gradient" but generalize to subgradients.

Algorithmic Category	Algorithm	Bound on Iterations		Cost per Iteration	Scales with Sparsity of $\mathbf{A}$	Relative Runtime Scaling: Ratio of Upper Bounds (Other Alg. / SCD)
		Lasso	Logreg			
Coordinate Descent	Cyclic CD	$d\|\mathbf{x}^*\|_2^2/\varepsilon$	$d\|\mathbf{x}^*\|_2^2/\varepsilon$	n	Yes	—
	Stochastic CD (SCD)	$d\|\mathbf{x}^*\|_2^2/\varepsilon$	$d\|\mathbf{x}^*\|_2^2/\varepsilon$	n	Yes	—
	Greedy CD	$\|\mathbf{x}^*\|_1^2/\varepsilon$	$\|\mathbf{x}^*\|_1^2/\varepsilon$	nd	Yes	$\|\mathbf{x}^*\|_0$
Iterative Shrinkage/ Thresholding	GPSR-BB	$?/\epsilon$	N/A	$k \cdot nd$	Yes	$? \cdot k$
	SpaRSA	$?/\epsilon$	$?/\epsilon$	$k \cdot nd$	Yes	$? \cdot k$
	FPC_AS	$?/\epsilon$	N/A	$k \cdot nd^2$	**No**	$? \cdot k \cdot d$
Compressed Sensing	Hard_l0	N/A	N/A	nd	Yes	N/A
	CoSaMP	N/A	N/A	nd	Yes	N/A
Homotopy	LARS	d	N/A	$d(d+n)$	**No**	$\varepsilon/\|\mathbf{x}^*\|_2^2 \cdot d^2$
Stochastic Gradient	Vanilla SGD	N/A	$\kappa_1\|\mathbf{x}^*\|_2^2/\varepsilon^2$	d	Yes	$1/n \cdot d/\varepsilon$
	TruncGrad	$\kappa_1\|\mathbf{x}^*\|_2^2/\varepsilon^2$	$\kappa_1\|\mathbf{x}^*\|_2^2/\varepsilon^2$	d	Yes	$1/n \cdot d/\varepsilon$
	SMIDAS	$\|\mathbf{x}^*\|_1^2/\varepsilon^2$	$\|\mathbf{x}^*\|_1^2/\varepsilon^2$	d	**No**	$1/n \cdot \|\mathbf{x}^*\|_0/\varepsilon$
	RDA	$c^2 \cdot d\|\mathbf{x}^*\|_2^2/\varepsilon^2$	$\kappa_1\|\mathbf{x}^*\|_2^2/\varepsilon^2$	d	**No**	$1/n \cdot d/\varepsilon$
Accelerated	FISTA	$\sqrt{\rho}\|\mathbf{x}^*\|_2/\sqrt{\varepsilon}$	$\sqrt{\rho}\|\mathbf{x}^*\|_2/\sqrt{\varepsilon}$	$k \cdot nd$	Yes	$\sqrt{\epsilon}/\|\mathbf{x}^*\|_2 \cdot k \cdot \sqrt{\rho}$
Interior Point	L1_LS, L1_logreg	$\kappa_3 \log(d/\varepsilon)$	$\kappa_3 \log(d/\varepsilon)$	nd^2	**No**	$\varepsilon/\|\mathbf{x}^*\|_2^2 \cdot d^{3/2}$
Distributed	ADMM	?	?	$k \cdot nd$	Yes	$? \cdot k$
	DDA	$\|\mathbf{x}^*\|_2^2 c^2 \rho^2/\varepsilon^2$	$\|\mathbf{x}^*\|_2^2 d/\varepsilon^2$	nd	Yes	d/ε

Notation: $c \doteq$ constant in Lasso constraint form in Eq. (4.12). $\rho \doteq \lambda_{max}(\mathbf{A}^T\mathbf{A})$. $\kappa_1 \doteq \max_i \|\mathbf{a}_i\|_2^2$. κ_3 depends on the log-barrier function multipliers in interior point. For algorithms without iteration bounds but with convergence rates, we write "?" to denote unknown constants. For algorithms with line searches, k is the number of line search iterations.

Table 4.1: **Related work, compared with** Stochastic CD (SCD)**.** (See Sec. 4.4.) **Upper Bound on Iterations**: Known bounds on the iterations required to achieve error at most $\varepsilon \doteq \frac{1}{n}\left[\mathbf{E}\left[F(\mathbf{x}^{(T)})\right] - F(\mathbf{x}^*)\right]$, ignoring small constants and log terms. **Cost per Iteration**: Approximate number of operations per iteration. **Scales with Sparsity of A** indicates if Cost per Iteration is proportional to the number of non-zero elements in **A**. In **Relative Runtime Scaling**, we compute each algorithm's runtime bound (Bound on Iterations × Cost per Iteration, choosing the better of Lasso or Logreg) and state its ratio with the SCD bound; blue terms indicate better scaling than SCD, and red terms indicate worse. As discussed in Sec. B.2, we approximate $\kappa_1 \approx n$, $\kappa_3 \approx \sqrt{d}$, and $\|\mathbf{x}^*\|_1^2 \leq \|\mathbf{x}^*\|_0 \|\mathbf{x}^*\|_2^2$.

of the classic algorithms from the statistics literature such as LARS [Efron et al., 2004], use a form of Greedy CD.

All three coordinate descent methods can take advantage of sparsity in $\mathbf{A}$. I.e., if only a fraction s of the entries of $\mathbf{A}$ are non-zero, then the algorithms' runtimes are multiplied by s.

Full gradient methods and Greedy CD generally require matrix-vector multiplications $\mathbf{Ax}$ on each iteration to compute the gradient, so they can benefit from pre-existing parallel linear algebra libraries. However, as we see in our experiments with L1_LS [Kim et al., 2007] and L1_logreg [Koh et al., 2007], the extra efficiency of coordinate descent can outweigh the benefit of parallel matrix-vector operations.

4.4.2 Batch vs. Stochastic Gradient

Both coordinate and full gradient methods can estimate the gradient using all or part of the training samples. *Batch gradient methods* compute the gradient using all of the samples. *Stochastic Gradient Descent (*SGD*)* attempts to speed up optimization by estimating the gradient on each iteration using a single randomly chosen sample. Many authors (e.g., Langford et al. [2009a]) have argued that SGD is one of the most scalable methods for large datasets, so we compare it in detail with SCD. Tab. 4.1 summarizes this comparison for four SGD variants: Vanilla SGD (a naive approach), Truncated Gradient (TruncGrad) [Langford et al., 2009a], Stochastic MIrror Descent Algorithm made Sparse (SMIDAS) [Shalev-Schwartz and Tewari, 2009], and Regularized Dual Averaging (RDA) [Xiao, 2010].

Though the upper bounds on runtime in Tab. 4.1 do not all have corresponding lower bounds, comparing the bounds for SCD with SGD variants lets us make several observations which are supported by our experiments:

- As the dimensionality d increases, SCD's performance should improve, relative to SGD. While the bounds for both methods explicitly increase as $O(d)$, the SGD bounds all include additional factors which can increase as $O(d)$.
- As the sample size n increases, SGD's performance should improve, relative to SCD. The total runtime bound for SGD is independent of n, while the runtime bound for SCD increases as $O(n)$.
- SCD will always overtake SGD in terms of the error ε after enough iterations. To see this, note that SCD requires $O(1/\varepsilon)$ iterations, while SGD requires $O(1/\varepsilon^2)$ iterations. Proponents of SGD sometimes argue that fast, approximate solutions suffice for machine learning, if the statistical error from using a finite sample size outweighs the approximation error [Bottou and Bousquet, 2007]. On the other hand, in our experiments, SCD overtakes SGD when ε is still quite large, sometimes from the beginning.

These dependencies on d, n, and ϵ listed above indicate that SCD and SGD perform best in different regimes. Below, we mention complications affecting SGD in all regimes.

Sparsity in $\mathbf{x}$: With SGD, it can be unclear how to balance the gradient of the loss, which is estimated very roughly, with the subgradient of the L_1 regularization term. Vanilla SGD produces dense weight vectors $x^{(t)}$, but most SGD algorithms use soft thresholding (Sec. 4.4.4) to shrink the weights towards zero. However, these sparsifying SGD methods often produce denser solutions than batch gradient methods [Shalev-Schwartz and Tewari, 2009, Xiao, 2010].

Sparsity in $\mathbf{A}$: Vanilla SGD and TruncGrad can both take advantage of sparsity in $\mathbf{A}$, though

TruncGrad must use a lazy update trick in Section 5 of Langford et al. [2009a]. The mirror update in SMIDAS and the averaging method in RDA prevent those methods from using lazy updates.

Termination criteria: Batch gradient methods often maintain all elements of the exact gradient, and comparing the gradient magnitude with a convergence tolerance parameter gives a theoretically sound and practical termination criterion. SGD uses very noisy estimates of the gradient, so it can be much more difficult to know when to declare convergence.

Some authors have proposed using mini-batches of samples in order to interpolate between batch and stochastic gradient methods. Zinkevich et al. [2010] and Dekel et al. [2012] present distributed SGD algorithms in which compute nodes handle different mini-batches, though their analyses do not handle non-smooth regularization (like L_1). We show empirical results for the algorithm from Zinkevich et al. [2010] in Sec. 4.5.2.

4.4.3 First-Order, Second-Order, and Accelerated Methods

First-order methods simply take steps in the direction of the gradient. Batch gradient computation generally results in $O(1/T)$ global convergence rates, while stochastic gradient results in $O(1/\sqrt{T})$ rates.

Adjusting the descent direction using the Hessian can improve the rate to $O(1/T^2)$. *Second-order methods* compute the Hessian directly, resulting in very expensive (but parallelizable) iterations, each of which cost $O(d^2)$ time. The interior point methods L1_LS [Kim et al., 2007] and L1_logreg [Koh et al., 2007] in Tab. 4.1 are second-order methods. Relative to first-order methods like SCD and Shotgun, interior point converges in very few (but costly) iterations. Since the interior point softens the L_1 constraints, it usually produces dense solutions $\mathbf{x}$, even when achieving near-optimal objective values.

Accelerated gradient methods, such as FISTA [Beck and Teboulle, 2009], implicitly estimate the Hessian using previous gradients, thereby achieving $O(1/T^2)$ convergence rates while maintaining iterations which cost $O(d)$ time. We list one accelerated method: Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [Beck and Teboulle, 2009]. FISTA accelerates traditional IST, achieving one of the best bounds in Tab. 4.1. Joseph B.: Say something about SpaRSA (and other IST-based methods?) doing better in our experiments. We'll need a good explanation...since that is strange.

Most accelerated and second-order methods update the full weight vector on each iteration. Nesterov [2010] show how to accelerate coordinate descent for unconstrained minimization of a strongly convex objective, which excludes our sparse regression problems. Extending the analysis to convex but not strictly convex objectives would be interesting, for it might lead to an accelerated version of SCD and Shotgun. However, their method requires dense vector updates, regardless of sparsity in $\mathbf{A}$ and $\mathbf{x}$.

4.4.4 Soft vs. Hard Thresholding

Naive optimization methods can produce dense approximate solutions $\mathbf{x}$, even while achieving near-optimal objective values. Several techniques help to generate sparse solutions $\mathbf{x}$.

Soft thresholding shrinks all weights $\mathbf{x}$ towards zero by a distance λ. It may be derived from penalty formulations, as in Eq. (4.2). The coordinate descent, stochastic gradient, and iterative shrinkage/thresholding methods in Tab. 4.1 all use soft thresholding (except for FPC_AS [Wen et al., 2010], which is designed for compressed sensing).

The same optimization problem may also be expressed in constraint form; for Lasso, the problem becomes:

$$\min_{\mathbf{x}\in\mathbb{R}^d} \tfrac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|_2^2 \quad \text{s.t.} \quad \|x\|_1 \le c\,. \tag{4.12}$$

The constant c may be chosen according to λ s.t. the penalty and constraint forms share the same optimum. Algorithms derived from the constraint form often use *gradient projection* to keep $\mathbf{x}$ in the feasible region. The projection can sparsify $\mathbf{x}$ by projecting some elements to zero. In Tab. 4.1, GPSR-BB uses gradient projection.

Constraint formulations with the L_1 norm replaced by the L_0 norm often result in algorithms which use *hard thresholding*, which sets all but a fixed number of elements of $\mathbf{x}$ to zero. In particular, compressed sensing algorithms, such as Hard_l0 [Blumensath and Davies, 2009] and CoSaMP [Needell and Tropp, 2010], often use hard thresholding. Likely because of this difference in problem formulations, in our experiments, Hard_l0 and CoSaMP were competitive with Shotgun on compressed sensing datasets but not on other types of data.

4.4.5 Parallel Algorithms

Parallel coordinate descent was also considered by Tsitsiklis et al. [1986], but for differentiable objectives in the asynchronous setting. They give a very general analysis, proving asymptotic convergence but not convergence rates. We are able to prove rates and theoretical speedups for our class of objectives.

Scherrer et al. [2012a,b], published after our initial work, discuss a general framework for parallel algorithms applicable to L_1-regularized loss minimization. They discuss SCD, Shotgun, and Greedy CD, as well as new parallel coordinate descent algorithms which choose groups of coordinates greedily or using graph colorings. Their convergence analysis and experiments indicate that careful clustering of coordinates by group can improve convergence.

Niu et al. [2011] present a method for parallelizing stochastic gradient which is similar in spirit to Shotgun. It makes parallel updates which can conflict, relying on the sparsity of the problem to limit the harmful impact of conflicts.

We also mention two very different methods for parallelizing L_1 regression, targeted at the distributed setting. The first is the Alternating Direction Method of Multipliers (ADMM) [Boyd et al., 2011b], which is applicable to master-slave systems which synchronize after every iteration. It is possible to parallelize optimization over samples or features [Boyd et al., 2011b], or even both at once [Parikh and Boyd, 2012]. To our knowledge, though ADMM converges [Boyd et al., 2011b], there are no known global convergence rates for ADMM w.r.t. its objective, though there are bounds w.r.t. related values [He and Yuan, 2012, Wang and Banerjee, 2012].

The second distributed method is Distributed Dual Averaging (DDA) [Duchi et al., 2012], which is a general method for distributed optimization. The method handles distributing over samples and features, asynchronous computation, and stochastic gradient estimates. Their method achieves slower convergence rates than SCD and Shotgun. Also, their bounds are w.r.t. the running average of iterates $\mathbf{x}^{(t)}$, which will not be sparse in general.

4.5 Experimental Results

We present an extensive study of Shotgun for the Lasso and sparse logistic regression. On a wide variety of datasets, we compare Shotgun with published state-of-the-art solvers. We also analyze speedup in detail in terms of Theorem 4.3.2 and hardware issues.

4.5.1 Lasso

We tested Shooting and Shotgun for the Lasso against five published Lasso solvers on 35 datasets. We summarize the results here; details are in the supplement.

Implementation: Shotgun

Our implementation made several practical improvements to the basic Shooting and Shotgun algorithms. We approximated a stochastic choice of $\mathcal{P}_t$ by cycling through the weights.

Following Friedman et al. [2010], we maintained a vector $\mathbf{Ax}$ to avoid repeated computation. We also used their *continuation* scheme: rather than directly solving with the given λ, we solved with an exponentially decreasing sequence $\lambda_1, \lambda_2, \dots, \lambda$. The solution $\mathbf{x}$ for λ_k is used to warm-start optimization for λ_{k+1}. The first regularizer λ_1 is chosen using λ_{max}, the minimum value for which the optimum x^* will be all zeros. Earlier sub-problems use looser termination criteria. This scheme, detailed in Alg. 4.3, can give significant speedups.

Though our analysis is for the synchronous setting, our implementation was asynchronous because of the high cost of synchronization. We used atomic compare-and-swap operations for updating the $\mathbf{Ax}$ vector.

We used C++ and the CILK++ library [Leiserson, 2009] for parallelism. All tests ran on an AMD processor using up to eight Opteron 8384 cores (2.69 GHz).

Algorithm 4.3: Shotgun for Lasso (continuation)

```
Input: Data $\mathbf{A} \in \mathbb{R}^{n \times d}$, $\mathbf{y} \in \mathbb{R}^n$; scalar $\lambda \geq 0$
Set $\mathbf{x} = \mathbf{0} \in \mathbb{R}^d_+$, $K = 1 + \lfloor d/2000 \rfloor$
Set $\lambda_{max} = \max |2(A^T y)_i|$, $\alpha = \left(\frac{\lambda_{max}}{\lambda}\right)^{1/K}$
foreach $k = K, K-1, \dots, 0$ do
    Set $\lambda_k = \lambda \alpha^k$
    Solve subproblem: $x =$Shotgun$(x, \lambda_k)$
```

Other Algorithms

We tested several of the algorithms discussed in Sec. 4.4:

- L1_LS [Kim et al., 2007]: log-barrier interior point. The implementation is in Matlab®, but the expensive step (PCG) uses very efficient native Matlab calls. In our tests, matrix-vector operations were parallelized on up to 8 cores.
- FPC_AS [Wen et al., 2010]: iterative shrinkage/thresholding.
- GPSR-BB [Figueiredo et al., 2008]: gradient projection.
- Hard_l0 [Blumensath and Davies, 2009]: iterative hard thresholding. We set its sparsity level for hard thresholding to be the sparsity obtained by Shooting.
- SpaRSA [Wright et al., 2009]: iterative shrinkage/thresholding.

As with Shotgun, all of Shooting, FPC_AS, GPSR-BB, and SpaRSA use continuation. Links to implementations of these algorithms are provided in the supplementary material.

We also tested published implementations of the classic algorithms GLMNET [Friedman et al., 2010] and LARS [Efron et al., 2004]. Since we were unable to get them to run on our larger datasets, we exclude their results.

Note: For the experiments in our conference paper, we did not column-normalize the data (divide each column in A by its Euclidian norm), which caused problems with some of the other algorithms. Column-normalizing the data requires a small change to the optimization problem: the penalty λ must be re-scaled for each coordinate. With this fix, we were able to get a modified SpaRSA implementation to solve some of the very large problems, and we have included these new results. However, Shotgun and Shooting are still much more efficient for these largest problems.

Results

We divide our comparisons into four categories of datasets. Some dataset statistics are listed in Tab. 4.2, and the supplementary material has more detailed descriptions.

Sparco: Real-valued datasets of varying sparsity and characteristics from the Sparco testbed [van den Berg et al., 2009].
$n \in [128, 29166]$, $d \in [128, 29166]$

Single-Pixel Camera: Dense compressed sensing problems from the Single-Pixel Camera project [Duarte et al., 2008].
$n \in [410, 4770]$, $d \in [1024, 16384]$

Sparse Compressed Imaging: Similar to Single-Pixel Camera datasets, but with very sparse random $-1/+1$ measurement matrices. Created by us.
$n \in [477, 32768]$, $d \in [954, 65536]$

Large, Sparse Datasets: Very large and sparse problems, including predicting stock volatility from text in financial reports [Kogan et al., 2009].
$n \in [30465, 209432]$, $d \in [209432, 5845762]$

We ran each algorithm on each dataset with regularization $\lambda = 0.5$ and 10. Fig. 4.3 summarizes runtime results, divided by dataset category, and Tab. 4.2 gives actual numbers for the most successful algorithms. We omit runs which failed to converge within a reasonable time period.

Shotgun (with $\texttt{P} = 8$) consistently performs well, converging faster than other algorithms on most dataset categories. Shotgun does particularly well on the Large, Sparse Datasets category, for which

Dataset	n	d	% non-z	λ	Shotgun P=8	Shooting	SpaRSA	L1_LS	GPSR_BB
sparco402	29166	86016	3.5	0.5	29.6s (285.2)	101.0s (285.2)	**25.6s** (285.4)	320.1s (285.2)	46.8s (285.3)
sparco402	29166	86016	3.5	10.0	4.3s (533.0)	17.3s (533.0)	**1.0s** (533.0)	24.4s (533.0)	**1.0s** (533.0)
sparco501	629	4096	92.0	0.5	1.3s (42.0)	3.5s (42.0)	0.8s (51.2)	13.9s (51.2)	**0.8s** (51.2)
sparco501	629	4096	92.0	10.0	**0.1s** (109.6)	0.2s (109.6)	0.2s (121.2)	0.8s (121.2)	0.2s (121.2)
sparco502	10000	10000	33.2	0.5	**8.5s** (113.3)	28.0s (113.3)	-	-	-
sparco502	10000	10000	33.2	10.0	**1.3s** (148.4)	5.5s (148.4)	-	-	-
sparco603	1024	4096	70.7	0.5	0.6s (89.7)	1.5s (89.7)	**0.3s** (89.8)	3.3s (89.8)	0.4s (89.8)
sparco603	1024	4096	70.7	10.0	0.3s (592.7)	0.6s (592.7)	**0.2s** (592.7)	1.2s (593.0)	0.4s (592.7)
Ball64_singlepix	1638	4096	50.0	0.5	-	**8.5s** (0.3)	-	89.8s (0.3)	287.3s (0.3)
Ball64_singlepix	1638	4096	50.0	10.0	4.0s (6.1)	**2.8s** (6.1)	286.6s (6.3)	26.1s (6.1)	287.4s (6.1)
Logo64_singlepix	1638	4096	50.0	0.5	**1.0s** (0.3)	6.7s (0.3)	202.4s (0.3)	57.8s (0.3)	280.6s (0.3)
Logo64_singlepix	1638	4096	50.0	10.0	3.1s (5.8)	**3.0s** (5.9)	-	25.7s (5.8)	288.0s (5.8)
Mug128_singlepix	4770	16384	50.0	0.5	-	**93.1s** (1.9)	-	1773.3s (1.9)	4333.4s (1.9)
Mug128_singlepix	4770	16384	50.0	10.0	4398.9s (37.7)	**122.1s** (37.7)	3222.0s (38.4)	411.5s (37.7)	3228.3s (38.1)
mug05_12_12	12410	24820	0.3	0.5	**0.7s** (1684.8)	1.1s (1684.8)	3.9s (1685.7)	14.6s (1684.8)	7.1s (1685.8)
mug05_12_12	12410	24820	0.3	10.0	**0.3s** (30353.9)	0.8s (30353.9)	0.7s (30354.0)	5.4s (30359.5)	0.7s (30354.0)
peppers025_12_12	16384	65536	0.2	0.5	**4.9s** (4140.1)	8.0s (4140.0)	19.5s (4164.7)	55.6s (4139.6)	43.0s (4165.0)
peppers025_12_12	16384	65536	0.2	10.0	**3.7s** (76191.3)	7.4s (76191.3)	**3.6s** (76191.9)	13.3s (76208.9)	7.7s (76192.3)
peppers05_12_12	32768	65536	0.1	0.5	**2.8s** (4606.4)	5.3s (4606.3)	12.4s (4607.2)	33.1s (4606.2)	23.7s (4607.8)
peppers05_12_12	32768	65536	0.1	10.0	**1.8s** (82701.4)	4.7s (82701.4)	2.1s (82701.4)	15.0s (82742.0)	2.1s (82701.4)
finance1000	30465	216842	0.1	0.5	**114.1s** (2881.6)	135.9s (2881.9)	3943.4s (2883.0)	7926.0s (2895.8)	-
finance1000	30465	216842	0.1	10.0	**33.1s** (26462.7)	53.6s (26461.5)	402.4s (26463.0)	2555.1s (26453.7)	-
financebigram	30465	5845762	< 0.1	0.5	**2005.6s** (321.0)	4920.0s (322.5)	-	-	-
financebigram	30465	5845762	< 0.1	10.0	**1841.0s** (4718.2)	5027.3s (4727.5)	-	-	-
phonecalls	209432	209432	2e-5	0.5	**141.5s** (231207.0)	1172.1s (220253.0)	708.0s (231206.0)	-	-
phonecalls	209432	209432	2e-5	10.0	**34.0s** (281732.0)	101.5s (280254.0)	382.0s (281762.0)	-	-

Table 4.2: A subset of the experiments for L_1-regularized least squares (Lasso). **Dataset Statistics**: n (number of samples), d (number of features), % non-zeros in **A**. λ: For each dataset we tested $\lambda = 0.5$ and $\lambda = 10.0$. **Results**: For each algorithm, the left column shows running time in seconds, and the right column shows the objective value achieved. The fastest qualified time is shown in bold face. Algorithms terminate at different objective values due to intrinsic differences in termination conditions. We only include results in which an algorithm came within 25% of the best objective found by any other algorithm. Joseph B.: Should we include ρ or $\mathtt{P}^*$ in this table?

most algorithms failed to converge anywhere near the ranges plotted in Fig. 4.3. The largest dataset, whose features are occurrences of bigrams in financial reports [Kogan et al., 2009], has 5 million features and 30K samples. On this dataset, `Shooting` converges but requires ~ 4900 seconds, while `Shotgun` takes < 2000 seconds.

On the Single-Pixel Camera datasets, `Shotgun` ($\mathtt{P} = 8$) is slower than `Shooting`. In fact, it is surprising that `Shotgun` converges at all with $\mathtt{P} = 8$, for the plotted datasets all have $\mathtt{P}^* = 1$. Fig. 4.2 shows `Shotgun` with $\mathtt{P} > 4$ diverging for the `Ball64_singlepixcam` dataset; however, after the practical adjustments to `Shotgun` used to produce Fig. 4.3, `Shotgun` converges with $\mathtt{P} = 8$.

Among the other solvers, `L1_LS` is the most robust and even solves some of the Large, Sparse Datasets.

It is difficult to compare optimization algorithms and their implementations. Algorithms' termination criteria differ; e.g., primal-dual methods such as `L1_LS` monitor the duality gap, while `Shotgun` monitors the change in $\mathbf{x}$. `Shooting` and `Shotgun` were written in C++, which is generally fast; the other algorithms were in Matlab, which handles loops slowly but linear algebra quickly. Therefore, we emphasize major trends: `Shotgun` robustly handles a range of problems; Theorem 4.3.2 helps explain its speedups; and `Shotgun` generally outperforms published solvers for the Lasso.

4.5.2 Sparse Logistic Regression

For logistic regression, we focus on comparing `Shotgun` with `Stochastic Gradient Descent (SGD)` variants. `SGD` methods are of particular interest to us since they are often considered to be very efficient, especially for learning with many samples; they often have convergence bounds independent of the number of samples.

For a large-scale comparison of various algorithms for sparse logistic regression, we refer the reader to the recent survey by Yuan et al. [2010]. On `L1_logreg` [Koh et al., 2007] and CDN [Yuan et al., 2010], our results qualitatively matched their survey. Yuan et al. [2010] do not explore `SGD` empirically.

Implementation: `Shotgun CDN`

Our version of `Shotgun` for sparse logistic regression is based on the sequential Coordinate Descent Newton (CDN) method proposed by Yuan et al. [2010]. Their algorithm performs cyclic coordinate descent on the L_1-regularized logistic loss function, but instead of first-order gradient descent, it uses the second-order Newton method. (There is no closed-form solution for coordinate minimization for logistic loss.) On iteration t, for each coordinate j, CDN solves a second-order approximation of the objective function F:

$$x_j^{(t+1/2)} \longleftarrow x_j^{(t)} + \min_z \left\{ |x_j^{(t)} + z| - |x_j^{(t)}| + F_j' z + \frac{1}{2} F_j'' z^2 \right\}, \tag{4.13}$$

where we abbreviate $F_j' \doteq \frac{\partial}{\partial x_j} F(x^{(t)})$ and $F_j'' \doteq \frac{\partial^2}{\partial x_j^2} F(x^{(t)})$. This equation has a closed form "soft-thresholding" solution:

$$x_j^{t+1/2} \longleftarrow x_j^{(t)} + S\left(1 - F_j', \, F_j'' x_j^{(t)}\right) / F_j'' \, , \tag{4.14}$$

where $S(y, h) = \mathrm{sign}(y)(|y| - h)_+$. To guarantee that the objective decreases, the new value $x_j^{(t)}$ is chosen via a standard backtracking line search which terminates once a sufficient decrease condition is reached. To improve performance, CDN maintains an *active set* of weights. Weights in the active set are allowed to become non-zero, but weights outside the set remain set to zero. We refer the reader to Yuan et al. [2010] for more details. We refer to this sequential algorithm as `Shooting CDN`.

`Shotgun CDN`, our parallel version of `Shooting CDN`, executes the coordinate minimizations in Eq. (4.14) in parallel. After each cycle of coordinate minimizations, the active set of weights is updated sequentially. Alg. 4.4 outlines the `Shotgun CDN` algorithm.

As Yuan et al. [2010] show empirically for sparse logistic regression, their CDN method is often orders of magnitude faster than the basic `Shooting` algorithm (Alg. 4.1), which uses a fixed step size and no active set. We note that our analysis uses a fixed step size and does not consider the effects of a line search. However, we find empirically that `Shotgun CDN` is an efficient and robust algorithm.

Algorithm 4.4: `Shotgun CDN`

```
Input: Data $\mathbf{A} \in \mathbb{R}^{n \times d}$, $\mathbf{y} \in \mathbb{R}^n$; scalar $\lambda \geq 0$
Set $\mathbf{x} = \mathbf{0} \in \mathbb{R}_+^d$
Set active-set $= \mathbf{x}$
while not converged do
    Assign weights in active-set to processors.
    In parallel on P processors do
        Get assigned weight $j$.
        Set $x_j^{t+1/2}$ as in Eq. (4.14).
        Execute back-tracking line search to set $x_j^{t+1}$.
    Update active-set by removing all coordinates $j$ s.t. $x_j^{(t)} = 0$ and $F_j'$ is small.
```

Other Algorithms

`SGD` iteratively updates $\mathbf{x}$ in a gradient direction estimated with one sample and scaled by a learning rate. We implemented `SGD` in C++ following, e.g., Zinkevich et al. [2010]. We used lazy shrinkage updates [Langford et al., 2009a] to make use of sparsity in $\mathbf{A}$. Choosing learning rates for `SGD` can be challenging. In our tests, constant rates led to faster convergence than decaying rates (decaying as $1/\sqrt{T}$). For each test, we tried 14 exponentially increasing rates in $[10^{-4}, 1]$ (in parallel) and chose the rate giving the best training objective. We did not use a sparsifying step for `SGD`.

`SMIDAS` [Shalev-Schwartz and Tewari, 2009] uses stochastic mirror descent but truncates gradients to sparsify $\mathbf{x}$. We tested their published C++ implementation.

`Parallel SGD` refers to the work by Zinkevich et al. [2010], which runs `SGD` in parallel on different subsamples of the data and averages the solutions $\mathbf{x}$. We tested this method since it is one of the few existing methods for parallel regression, but we note that Zinkevich et al. [2010] did not address L_1 regularization in their analysis. We averaged over 8 instances of `SGD`.

See Sec. 4.4 for more information on these algorithms.

Results

Fig. 4.4 plots training objectives and test accuracy (on a held-out 10% of the data) for two large datasets.

The zeta dataset [4] illustrates the regime with $n \gg d$. It contains 500K samples with 2000 features and is fully dense in $\mathbf{A}$. SGD performs well and is fairly competitive with Shotgun CDN (with $\mathtt{P} = 8$).

The rcv1 dataset [5] [Lewis et al., 2004] illustrates the high-dimensional regime ($d > n$). It has about twice as many features (44504) as samples (18217), with 17% non-zeros in $\mathbf{A}$. Shotgun CDN ($\mathtt{P} = 8$) was much faster than SGD, especially in terms of the objective. Parallel SGD performed almost identically to SGD.

Though convergence bounds for SMIDAS are comparable to those for SGD, SMIDAS iterations take much longer due to the mirror descent updates, which require $O(d)$ exponentiations. To execute 10M updates on the zeta dataset, SGD took 728 seconds, while SMIDAS took over 8500 seconds.

Note that Fig. 4.4 confirms the predictions in Sec. 4.4.2 about SCD and SGD's relative performances within different regimes. In the high-n regime (zeta), SGD initially outperforms SCD (Shooting CDN) but is then overtaken. In the high-d regime (rcv1), SCD outperforms SGD.

These results highlight how SGD is orthogonal to Shotgun: SGD can cope with large n, and Shotgun can cope with large d. A hybrid algorithm might be scalable in both n and d and, perhaps, be parallelized over both samples and features.

4.5.3 Speedup of Shotgun

To study the speedup of Shotgun Lasso and Shotgun CDN, we ran both solvers on our datasets with varying λ, using varying P (number of parallel updates = number of cores). We recorded the running time as the first time when an algorithm came within 0.5% of the optimal objective, as computed by Shooting.

Fig. 4.5 shows results for both speedup (ratio of sequential/parallel running time) and iteration speedup (ratio of sequential/parallel iterations until convergence). The iteration speedups match Theorem 4.3.2 quite closely. However, relative speedups in iterations (about $8\times$) are not matched by speedups in running time (about $2\times$ to $4\times$). Running more updates in parallel slows down each update.

We thus discovered that speedups in time were limited by low-level technical issues. To understand the limiting factors, we studied how the relative parallel performance was affected by artificial modifications to the algorithm code. We list two important tests here and refer the reader to the supplement for more details.

First, recall that each weight update requires an atomic update to the shared $\mathbf{Ax}$ vector. We tested a version of the algorithm which did not protect the shared $\mathbf{Ax}$ vector from concurrent modifications, but this change did not make iterations run much faster, indicating that atomic updates did not harm performance much.

Second, we increased the computational complexity of the algorithm by evaluating a trigonometric function several times for each weight update and noticed that the parallel speedup improved very quickly when the computation per update was increased. This result indicated that the ratio of floating point operations to memory accesses was too low. Since each update uses a different column of $\mathbf{A}$, data accesses have no temporal locality, so this ratio is only $O(1)$.

[4] The zeta dataset is from the Pascal Large Scale Learning Challenge: http://www.mlbench.org/instructions/

[5] Our version of the rcv1 dataset is from the LIBSVM repository [Chang and Lin, 2001].

Our experiments showed that the parallel performance was limited by us hitting the *memory wall* [Wulf and McKee, 1995]: memory bus bandwidth and latency proved to be the most limiting factors. We further validated these conclusions by monitoring hardware performance counters provided by the Linux performance monitoring interface. The counters showed that the amount of CPU cache misses was very large in comparison to the mathematical instructions executed.

4.6 Discussion

We introduced `Shotgun`, a simple parallel algorithm for L_1-regularized loss minimization. Our convergence results for `Shotgun` are the first such results for parallel coordinate descent with L_1 regularization. Our bounds predict near-linear speedups, up to an interpretable, problem-dependent limit. In experiments, these predictions matched empirical behavior.

Extensive comparisons showed that `Shotgun` outperforms state-of-the-art L_1 solvers on many datasets. Though hardware constraints somewhat limited speedups, future hardware and algorithmic improvements may permit even greater scaling. We believe that, currently, `Shotgun` is one of the most efficient and scalable solvers for L_1-regularized problems.

Code, Data, and Benchmark Results: `http://www.select.cs.cmu.edu/projects`

4.7 Future Work

Our initial work on `Shotgun` demonstrated a method for exposing the potential for multicore parallelization inherent in a certain class of optimization problems. We believe our ideas can be used much more generally. In this section, we propose extensions to (a) the algorithm, to maximize benefits from parallelization (Sec. 4.7.1), (b) the analysis, to discover other model classes for which `Shotgun` is applicable (Sec. 4.7.2), and (c) the computing platform, to explore other platforms with alternative hardware constraints (Sec. 4.7.3 and Sec. 4.7.4).

4.7.1 Generalized `Shotgun` Algorithm

An adaptive version of `Shotgun` might permit greater parallelism or faster convergence. For example, after computing updates in parallel, an additional step could re-weight updates to eliminate harmful conflicts. Another type of adaptivity could involve pre-processing to optimize the order in which coordinates are updated. Some similar ideas are explored by Scherrer et al. [2012a,b], who discuss an extra step for choosing updates to commit, as well as pre-processing to find blocks of highly correlated features.

The Shotgun algorithm itself could also be generalized to interpolate between coordinate descent and full-gradient descent in order to trade off total computation and parallelism. As noted in Sec. 4.3, our analysis of `Shotgun` involving `P` (the number of coordinates updated in parallel) indicates that we can increase `P` and decrease the (fixed) step size while still guaranteeing convergence. `Shotgun` represents one extreme of this spectrum, with a smaller number of coordinate updates (providing less opportunity for parallelization) but a larger step size (resulting in less total computation empirically). At the other extreme, we could update all coordinates at once while decreasing the step size by $\mathtt{P}/d$ (without duplicated features). Understanding this trade-off would let us automatically tailor `Shotgun` to different platforms, from CPUs with few cores to Graphics Processing Units (GPUs) with many cores.

4.7.2 Analysis for Other Models

There are many learning problems related to the Lasso and sparse logistic regression, such as the graphical Lasso [Friedman et al., 2008], which have similar but more complicated structures. Generalizing `Shotgun` to objectives with more structure would be useful in practice and would shed light on how structure affects the inherent parallelism within problems. Also, more complex objectives requiring more operations per value loaded from memory could help to hide memory latency, giving better speedups for `Shotgun`.

4.7.3 `Shotgun` on Graphics Processing Units (GPUs)

General-purpose Graphics Processing Units (GPUs) are very cheap and powerful platforms for parallel computing, and they are rapidly developing in both speed and support for more general computation. To achieve max throughput, however, GPU code must be carefully designed to use the Single Instruction Multiple Data (SIMD) architecture. Our initial tests indicate potential for success but also the need for careful engineering to take full advantage of the hundreds of cores on modern GPUs.

Several recent works in machine learning have used GPUs for applications such as object detection [Coates et al., 2009], deep belief networks [Raina et al., 2009], and belief propagation [Grauer-Gray et al., 2008]. The largest speedups are achieved when the learning problem decomposes according to the GPU architecture; e.g., object detection works on images using operations similar to those for which GPU architecture was originally developed.

Background: GPU Architecture and CUDA

We give a very brief overview of NVIDIA GPU architecture and the NVIDIA CUDA extensions to the C language. Other GPU architectures, such as those from ATI, are somewhat different but are still optimized for graphics processing (e.g., vector/matrix operations).

Our initial experiments used a somewhat outdated NVIDIA Tesla C1060 GPU. The C1060 has 240 cores, called streaming processors. These cores are divided into sets of 8, each of which forms a streaming multiprocessor (SMP). Within a SMP, the 8 cores must operate in lockstep, executing the same instruction; branch divergence is handled by serializing the branches and so is very expensive. In theory, the C1060 can achieve about 933 GFLOPS.

A CUDA program running on a GPU is called a *kernel*, and it can consist of tens of thousands of *threads*. Threads are grouped into several *thread blocks*. Each thread block runs on a single SMP, and within that block, threads can communicate relatively cheaply. Threads in different blocks can only communicate via very expensive synchronizations. When a kernel is run, CUDA gives very few guarantees on the order in which blocks are executed and on the order in which threads within a block are executed. These constraints permit very fast execution for embarrassingly parallel applications but place significant restrictions on the types of computation a kernel can perform.

The Tesla C1060 memory hierarchy contains several levels of memory, the most important of which are *global memory* and *shared memory*. Global memory is the largest (about 4 GB total) and slowest, is accessible to all threads, and is optimized for coalesced accesses. Shared memory is small (about 16 KB per SMP) but very fast, and each thread block is given a separate space in this memory. A CPU thread can read from and write to the GPU global memory, but these operations are very expensive.

To achieve optimal performance, an application must avoid branch divergence within thread blocks, use coalesced accesses to global memory, minimize use of registers, and use thousands of threads.

Initial Tests

We have run initial tests of Shotgun on the GPU which indicate that Shotgun can indeed be extended to the GPU. However, we will need to do significant tuning to ensure efficient use of GPU memory. Our implementation of Shotgun on the GPU differs from that on the CPU in several aspects. GPU Shotgun operates synchronously (while the CPU version is asynchronous) and thus better matches the analysis in Sec. 4.3.1. GPU Shotgun uses multiple threads per coordinate update (while the CPU version uses one thread per coordinate update); the number of threads varies based on the sparsity of the variable's column in the design matrix $\mathbf{A}$ and can be as large as one thread block.

Our currently unimpressive results for GPU Shotgun can largely be explained in terms of memory latency and load balancing. GPUs require a certain number of FLOPs per memory access in order to hide memory latency (about 10 FLOPs per float on the Tesla C1060, as a general rule of thumb). For Shotgun, too few operations are required for each variable update to hide latency in our current implementation. Results from profiling GPU Shotgun indicate that actual memory bandwidth usage is generally far from the maximum possible.

In many of the sparse datasets we have tested, the sparsity of columns of design matrices $\mathbf{A}$ varies greatly, with some columns having many more non-zero elements than average. These denser columns take a disproportionate amount of the running time, sometimes up to about 95%. Better load balancing, perhaps by using multiple thread blocks for one dense column, could alleviate this problem.

We have tested several other optimizations which might prove useful for GPU Shotgun, including block coordinate descent, choosing coordinates to update based on a graph coloring (rather than stochastically), and handling dense columns in $\mathbf{A}$ sequentially.

4.7.4 `Shotgun` in the Distributed Setting

We have also considered extensions of `Shotgun` and other sparse regression algorithms to the distributed setting, in which communication constraints become a major bottleneck. Based on our initial experiments, we believe that a simple adaptation of `Shotgun` is impractical, for coordinate updates require far too little computation to hide communication latency.

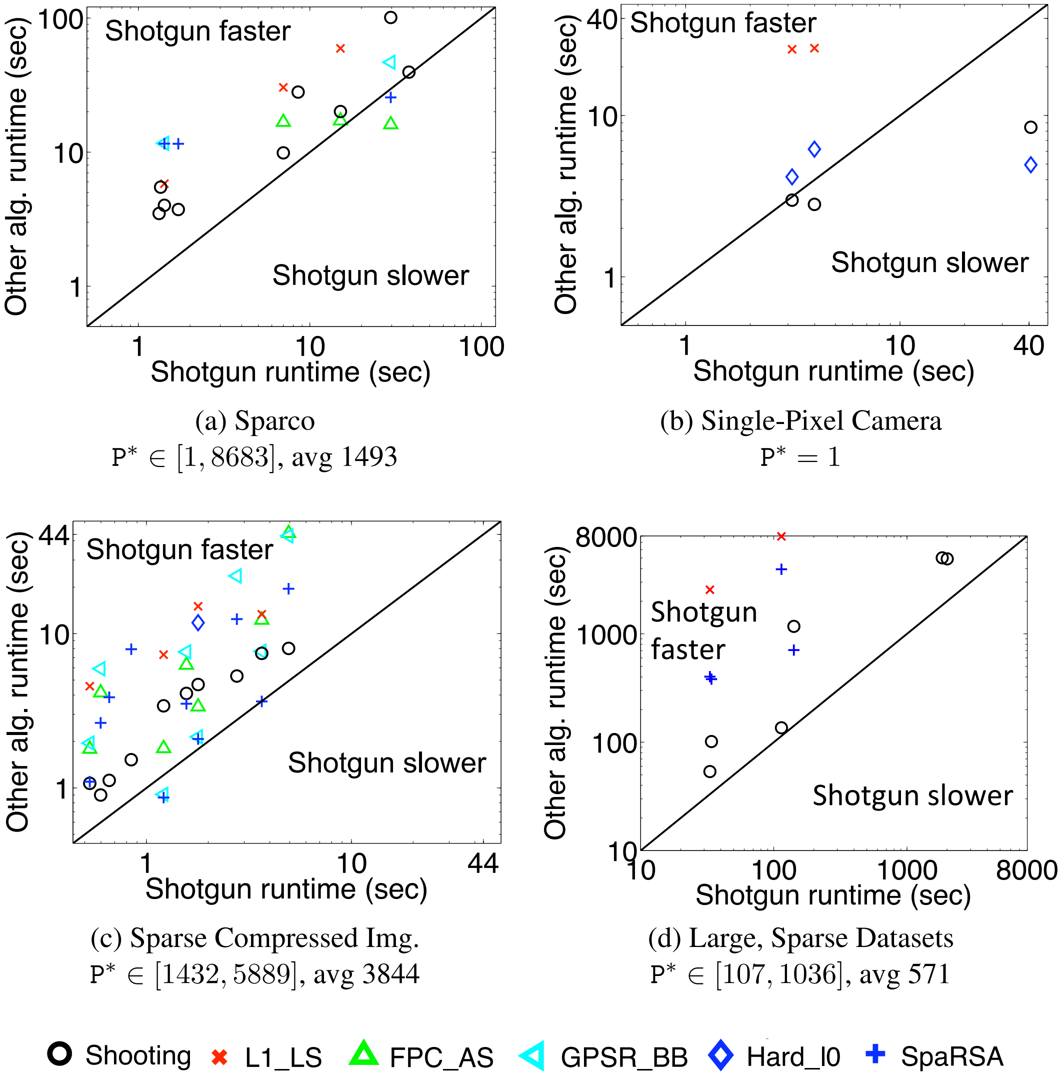
We list two ideas which look promising based on initial theoretical exploration and experiments with simulated parallel execution.

- Hybrid `Shotgun` algorithms: We could combine an existing distributed algorithm with `Shotgun`; the distributed algorithm would take advantage of distributed parallelism, and `Shotgun` would use multicore parallelism on each compute node. Many distributed algorithms split examples and/or features among compute nodes, each of which has a local subproblem which is a sparse regression-style problem (solvable using `Shotgun`). Some potentially useful distributed algorithms include:

  - `Parallel SGD`, such as the method analyzed by Zhang et al. [2012], Zinkevich et al. [2010], splits examples across compute nodes. Existing analysis should be easy to apply, where we replace `SGD` with `Shotgun` on each compute node.
  - `ADMM`: Alternating Directions Method of Multipliers [Boyd et al., 2011a, Parikh and Boyd, 2012], discussed in Sec. 4.4.5, may split examples and/or features across compute nodes.

- Sparsifying communication: Methods such as `ADMM` which iteratively communicate between compute nodes generally achieve better results and guarantees than methods such as Zinkevich et al. [2010] which use no communication (except at the end). However, `ADMM` and most similar methods for sparse regression require communicating dense vectors of total length $O(d)$ (number of features) or $O(n)$ (number of examples), limiting their scalability. Modifying `ADMM` to use sparse communication is non-trivial but could permit much greater scaling. One possible method would be to distribute the sparsifying L_1 penalty term, but this modification can make it difficult to prove convergence. Another possibility is to communicate changes in vectors rather than the vectors themselves, as well as applying truncation to promote sparsity; balancing error and sparsity introduced by truncation would require careful analysis.

Joseph B.: I removed discussion of GPU GraphLab.



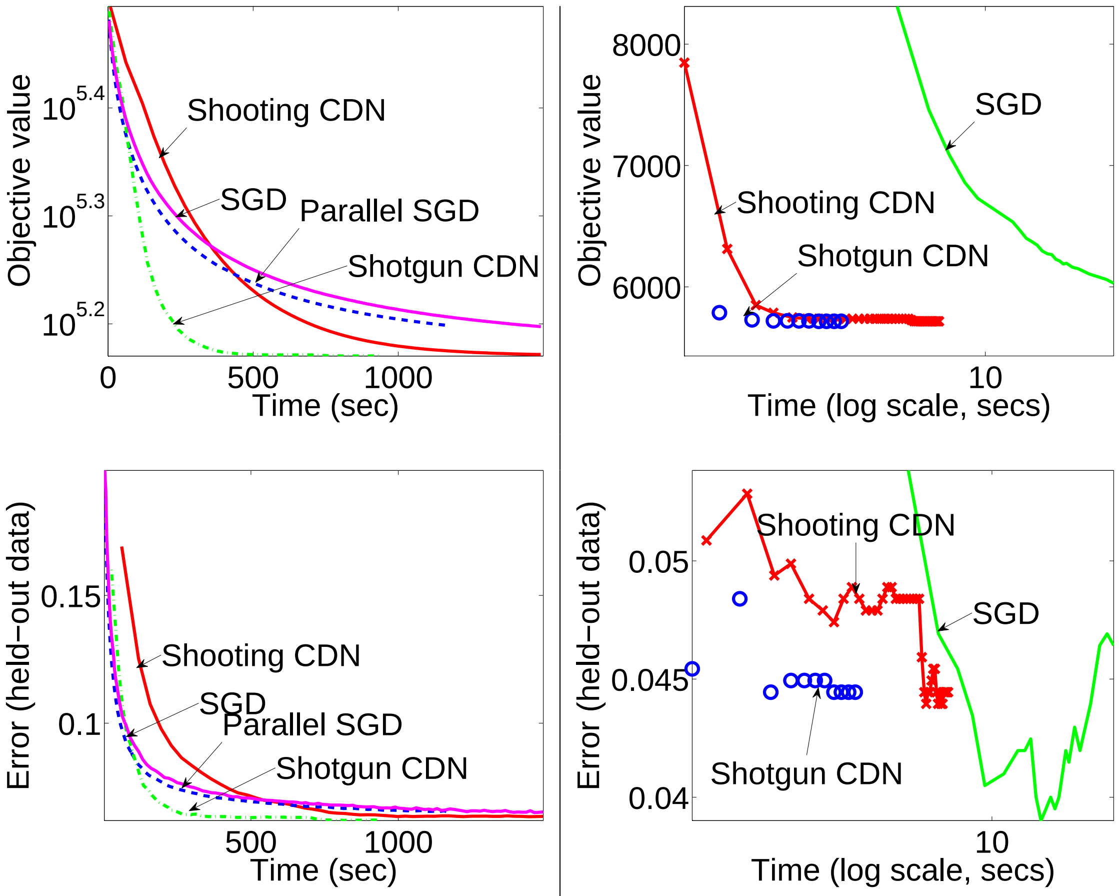
(a) Sparco
$\mathtt{P}^* \in [1, 8683]$, avg 1493

(b) Single-Pixel Camera
$\mathtt{P}^* = 1$

(c) Sparse Compressed Img.
$\mathtt{P}^* \in [1432, 5889]$, avg 3844

(d) Large, Sparse Datasets
$\mathtt{P}^* \in [107, 1036]$, avg 571

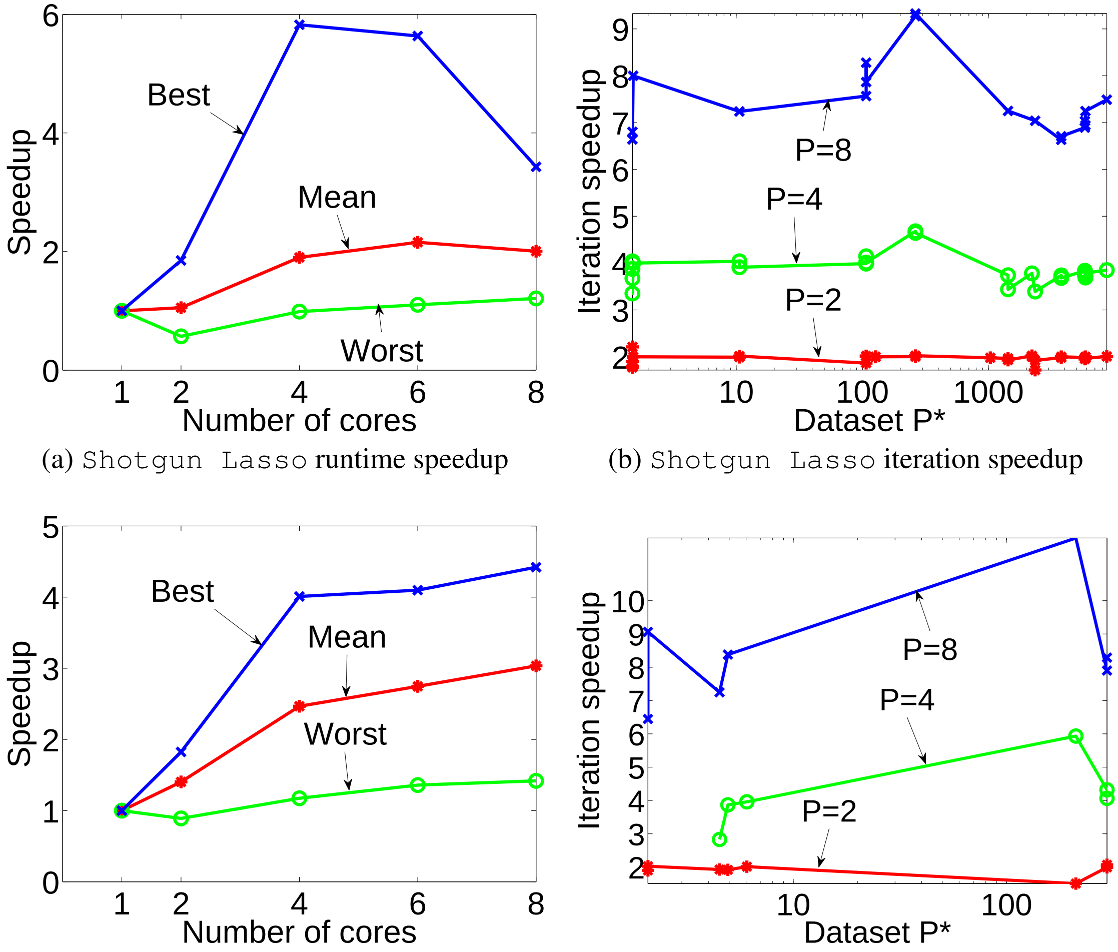
Figure 4.3: **Runtime comparison of algorithms for the Lasso on 4 dataset categories.** Each marker compares an algorithm with Shotgun (with $\mathtt{P} = 8$) on one dataset and $\lambda \in \{0.5, 10\}$. The Y-axis is that algorithm's running time; the X-axis is Shotgun's (P=8) running time on the same problem. Markers above the diagonal line indicate that Shotgun was faster; markers below the line indicate Shotgun was slower.



`zeta` with $\lambda = 1$
$d = 2000, n = 500,000$

`rcv1` with $\lambda = 1$
$d = 44504, n = 18217$

Figure 4.4: **Sparse logistic regression on 2 datasets.** The top plots trace training objectives over time; the bottom plots trace classification error rates on held-out data (10%). On `zeta` ($n \gg d$), `SGD` converges faster initially, but `Shotgun CDN` (P=8) overtakes it. On `rcv1` ($d > n$), `Shotgun CDN` converges much faster than `SGD`; `Parallel SGD` (P=8) is hidden by `SGD`. (Note the log scale for `rcv1`.) Joseph B.: Standardize colors in plots. Joseph B.: Do cross-val to choose lambdas.



(a) `Shotgun Lasso` runtime speedup

(b) `Shotgun Lasso` iteration speedup

(c) `Shotgun CDN` runtime speedup

(d) `Shotgun CDN` iteration speedup

Figure 4.5: **Speedup analysis.** **(a,c)** Speedup in running time for `Shotgun Lasso` and `Shotgun CDN` (sparse logistic regression). Mean, min, and max speedups over datasets from Sec. 4.5. **(b,d)** Speedup in iterations until convergence as a function of `P`*, for each dataset in Sec. 4.5. Both `Shotgun` instances exhibit almost linear speedups w.r.t. iterations. Joseph B.: We should make the y-axes the same for Lasso and logreg.

Chapter 5

Conclusions

This thesis has examined Conditional Random Fields, one of the most flexible and empirically successful classes of probabilistic models. We argued that, while traditional learning methods often scale poorly on large problems, we can achieve much greater scaling using alternative learning methods based on two guiding principles: *decomposition* and *trade-offs*. Decomposition refers to simplifying a learning problem by breaking it down into smaller, easier subproblems. Trade-offs refers to balancing sample complexity, computational complexity, and potential for parallelization, one of which may often be traded for another.

We proved this thesis statement by developing novel techniques for parameter learning, structure learning, and regression. In Ch. 2, we examined two parameter learning methods, composite likelihood (and the special case of pseudolikelihood) and the canonical parameterization of Abbeel et al. [2006]. Both methods decompose the learning problem into many smaller subproblems of regressing one (or a subset) of the variables on its Markov blanket. Our empirical results show how this decomposition permits much faster learning than the original problem (maximum-likelihood estimation). Moreover, our analysis of composite likelihood in particular reveals a large class of models which are efficiently (PAC) learnable. Our bounds indicate an enormous potential for optimizing trade-offs in learning. Composite likelihood estimator structure may be chosen to achieve low sample complexity while simultaneously maintaining tractable computation (inference) and simple parallelization. In addition, the choices of joint and disjoint optimization allow trade-offs between sample complexity and parallelism.

In Ch. 3, we presented initial results applying these themes to structure learning. Our method, based on the Chow-Liu algorithm for MRFs [Chow and Liu, 1968], primarily spends computation on edge weights, which may be computed independently for every possible edge in the model. This decomposition makes the task of structure learning mostly data parallel. While we do not yet have strong theoretical guarantees for this method, we showed it to be empirically useful, and we detailed several paths for improvement.

In Ch. 4, we discussed regression, one of the key components of our parameter and structure learning methods. We proposed the `Shotgun` algorithm, a simple parallel coordinate descent method. Our method decomposes a multi-coordinate update into disjoint coordinate updates. As we increase the number of parallel updates, we risk more conflicts (resulting in more total computation required for convergence) but gain in parallelism. We demonstrated theoretically and empirically that it is easy to set the number of parallel updates to ensure that this trade-off is near-optimal; thus, `Shotgun` achieves near-linear speedups, though it was hampered by hardware constraints in our tests. Nevertheless, our extensive experiments showed `Shotgun` to be one of the most scalable existing algorithms for sparse regression.

In the following sections, we discuss two major ideas which integrate with our themes of decomposition and trade-offs. The first idea is taking advantage of *model structure and locality* in order to decompose learning problems (Sec. 5.1). The second idea is developing methods which *automatically adapt to the learning problem*, allowing improved trade-offs in learning (Sec. 5.2). These ideas generalize beyond our problems, and we believe that they will become increasingly important in machine learning.

We end by presenting a roadmap for learning MRFs and CRFs (Sec. 5.3). Our roadmap serves two purposes. First, we provide a guide for the practitioner who needs to figure out the best methods for his learning problem. Second, we discuss areas in need of improvement and with open questions. This discussion helps to place our CRF learning methods within a broader context, indicating both the areas in which our methods apply and the areas in which other methods might be preferable.

5.1 Model Structure and Locality

Our methods all attempt to simplify learning by using the structure and locality properties of the model being learned. Locality properties permit us to decompose (and simplify) learning problems.

In parameter learning, the methods we consider—pseudolikelihood, composite likelihood, and the canonical parameterization—are essentially regressing one (or a subset) of variables on the rest of the model. Each of these regression problems is defined according to the model parameterization and structure: only local factors and neighboring variables are involved in the regression. Moreover, these methods essentially work when *influence is local* within the model. That is, regressing one variable on its neighbors (in pseudolikelihood) gives a reasonable estimate of the local parameters when far-away parts of the model have little influence on that variable.

Similarly, our method for structure learning is based on the idea of local influence. Our Score Decay Assumption (Sec. 3.2.2) encodes this idea: scores, which measure influence between variables, should decay with distance in a model. We also use the idea of *evidence locality*, where direct interactions between output and input variables (via factors) are limited by restricting the number of input variables participating in each factor.

For parallel regression, our `Shotgun` algorithm is able to perform independent coordinate updates because of problem-specific limits on the interaction between features. This concept of locality is different than the graph-structured concept of locality used for CRF parameter and structure learning: even if a feature interacts with every other feature, limits on interaction strength can still isolate that feature.

Other researchers have also discussed decomposing problems based on concepts of structure and locality, especially for the sake of parallelization. For example, GraphLab [Low et al., 2010] uses the computational structure of problems for parallelization. Other works, such as recent analyses of parallel stochastic gradient descent [Zhang et al., 2012, Zinkevich et al., 2010], use statistical analyses to show how to parallelize problems. We believe our work on CRF parameter learning is one of the first efforts at taking advantage of both computational and statistical structure to permit parallel optimization. Future work on scalable machine learning will undoubtedly require similar joint analyses of both types of structure.

5.2 Model- and Data-Specific Methods

All of our work indicates the importance of designing methods which adapt to the model being learned or the training data. Adaptation can often be phrased as focusing learning on the parts of the problem which are hardest.

In our work on parameter learning, the success of composite likelihood was due to how the structure of composite likelihood estimators could conform to model structure and the correlations evident in the training data. Methods such as pseudolikelihood and the canonical parameterization are much less structured, so they cannot adapt as well to the model or data, hurting their performance. While pseudolikelihood and the canonical parameterization make the same approximations across the entire model, composite likelihood estimators can be structured to make fewer approximations in more important areas.

In our structure learning method, certain edge weights tended to work well on particular models (e.g., DCI with weaker $Y-Y$ factors and local CMI with weaker $Y-X$ factors). Our results indicate that, for our edge weight-based learning method to succeed in general, the method will need to choose an edge weight according to correlations in the data.

In our work on parallel regression, the number of parallel updates must be chosen according to the data. Certain datasets permit many parallel updates, while others require sequential updates. The extensions to our work in Scherrer et al. [2012a,b] (discussed in Sec. 4.4.5) further tailor learning to each dataset, improving the speed of convergence. Their work explicitly separates coordinates into correlated and uncorrelated groups (which are hard and easy, respectively, to optimize in parallel).

Other researchers have made related observations, often phrased in terms of focusing learning (or inference) on harder parts of models. For example, prediction cascades [Viola and Jones, 2001, Weiss and Taskar, 2010] use rough, simple models for classifying easy examples and finer, more complex models for hard examples. Splash sampling [Gonzalez et al., 2011] focuses inference on boundary regions in graphical models for image denoising. However, we believe that our work is one of the first to present a method which uses adaptation to optimize all three criteria of sample complexity, computational complexity, and potential for parallelism.

5.3 Roadmap for Learning MRFs and CRFs

This section presents a roadmap for practitioners for learning MRFs and CRFs. Sec. 5.3.1 compares parameter learning methods. Sec. 5.3.2 augments this discussion with lists of questions and answers to help practitioners identify the important aspects of their learning problem and choose the appropriate parameter learning method for their setting. Likewise, Sec. 5.3.3 compares structure learning methods, and Sec. 5.3.4 helps guide practitioners in choosing a method for their problem.

These discussions highlight many gaps in the current literature. Many algorithms have been proposed, but too few comparisons exist to state which methods are best. Our discussion of parameter learning is heavily biased towards the methods discussed in this thesis; we hope to do more comparisons with other methods in the future. Structure learning remains a particularly difficult problem, with many open questions.

5.3.1 Parameter Learning Methods

We divide our discussion of parameter learning methods into three parts: core methods, inference, and optimization. This division is not exact, for some methods in the literature are designed for particular types of inference or optimization. However, as we mention in Sec. 6.1, we hope that the field develops towards interchangeable parts so that the practitioner may select a core method, a type of inference, and an optimization method which work together well and fit the practitioner's learning problem.

Core methods

- *Supervised methods*
  - *Methods in this thesis*: Maximum likelihood, composite likelihood, and pseudolikelihood estimation (MLE, MCLE, MPLE, respectively)
    - These methods refer to the loss used; e.g., "maximum likelihood" refers to the log loss over the full model.
    - As shown in Ch. 2, MLE, MCLE, and MPLE form a range of methods (in that order). MLE makes the most of the data (achieving the best sample complexity or estimation error) and is the best with misspecified models, but it requires the most computation. MPLE is the opposite, and MCLE spans the range in between.
  - *Other scalable optimization-based methods*: Piecewise likelihood [Sutton and McCallum, 2005], Contrastive divergence [Hinton, 2002], SampleRank [Wick et al., 2011], Score matching [Hyvärinen, 2005], Ratio matching [Hyvärinen, 2007]
    - To our knowledge, no works have done broad analytical or empirical comparisons of these many other scalable methods with MLE, MCLE, and MPLE. A broad comparison would be invaluable for understanding how these methods relate to each other and which methods are best in different learning settings.
- *Semi-supervised methods*: posterior regularization [K. Ganchev and Taskar, 2010], generalized expectation [McCallum et al., 2007], expectation-maximization [Baum et al., 1970, Dempster et al., 1977]

Note: The above list does not include the matching-based methods discussed in Sec. 6.1 (the method of moments [Cam, 1986] and the canonical parameterization (Sec. 2.6)). While these methods show promise (e.g., method of moments for probabilistic topic models [Anandkumar et al., 2012]), we are unaware of large-scale comparisons showing these methods to be empirically competitive with optimization-based methods.

Inference

- *Exact*: Exact inference is ideal in terms of accuracy but generally intractable. Interesting work has been done on speeding up exact inference with arithmetic circuit representations (e.g., Darwiche [2003], Lowd and Domingos [2008]), as well as formulations as min-sum matrix products [Felzenszwalb and McAuley, 2011].
- *Approximate*: We list some sampling and variational inference methods. Both have been used successfully for fast learning, and it is unclear which is best in different learning settings.

  - *Sampling*: Gibbs sampling [Geman and Geman, 1984], structured Gibbs sampling [Asuncion et al., 2010], and more general Markov Chain Monte Carlo methods
    - In general, sampling methods are unbiased (i.e., compute the correct marginals), but only in the infinite-sample limit.
    - The stochasticity introduced by sampling can be useful when the loss is not convex, as in deep learning (e.g., [Le et al., 2011]).
  - *Variational*: Belief propagation [Pearl, 1988], structured versions of belief propagation (e.g., Gonzalez et al. [2009], Wainwright et al. [2005]), and other variational methods
    - In general, variational inference is biased, but it can converge quickly and can come with run-time guarantees on accuracy.
    - Since variational inference may be phrased as (sometimes convex) optimization, ideas and tools from the optimization literature can speed up inference (e.g., Hoffman et al. [2012]).

Optimization

- *Batch vs. stochastic*: Stochastic gradient methods are most useful with large training sets.
- *Coordinate vs. full gradient*: Coordinate descent is primarily useful with decomposable objectives, such as pseudolikelihood and composite likelihood, and with some types of variational inference which effectively decompose the objective (e.g., Hoffman et al. [2012]). When (exact or approximate) inference is run over the full model, the cost of inference is often large enough to outweigh the remaining computation required to compute the full gradient.
- *First-order vs. second-order*: First-order methods tend to be simpler to implement, but second-order methods, especially ones which approximate the Hessian (e.g., Liu and Nocedal [1989]), can be more efficient. Since inference is often the bulk of the computation in learning, the extra computation required by second-order methods may be minor.

Note: Hybrid methods which combine multiple techniques can sometimes take better advantage of each technique's optimal learning regime. For example, using pseudolikelihood (optimized only partway, not to completion) to initialize a max likelihood solution can be useful: pseudolikelihood can find a rough estimate of parameters quickly, avoiding many expensive iterations of max likelihood. Likewise, optimization with stochastic gradient descent can reach a rough approximation quickly, and a batch second-order method (with fast local convergence) can be used to reach the optimum.

5.3.2 Guide for Practitioners: Parameters

We divide this guide according to aspects of the learning problem: the model, the data, and computational constraints. We point out a few open questions indicating gaps in the literature.

Model

- *Is it a CRF or an MRF? Does the model permit tractable inference?* (Does it have low treewidth?)
  - Tractable MRF: Use MLE (with exact inference). In a junction tree representation, parameters may be learned without an iterative optimization procedure.

    - Intractable MRF: Two possible options are: (a) optimize log loss with the same approximate inference method which will be used at test time and (b) use structured composite likelihood. Option (a) is indicated by Wainwright [2006], but it is unclear if the intuition from that paper remains correct in other learning problems. Option (b) can be more efficient, but it remains unclear what type of inference is best at test time after composite likelihood training.
    - Tractable CRF: For fast training, use MLE (with exact inference) optimized via stochastic gradient descent. The approximate solution from stochastic gradient may be refined by switching to batch gradient descent in a two-stage optimization.
    - Intractable CRF: Use structured composite likelihood. For faster training, use stochastic gradient to optimize each composite likelihood component.

- *Is the model well-specified?* (I.e., does the "true" distribution follow the structure or class of structures which will be used for learning?)
    - If the model is not well-specified, then optimizing log loss (instead of pseudolikelihood or composite likelihood) is best when computationally feasible, for log loss has the lowest approximation error Liang and Jordan [2008].
- *Is the model parameterization convex?*
    - If the optimization problem is non-convex, then methods which introduce stochasticity in learning, such as stochastic gradient descent and sampling-based inference, may be useful in avoiding local minima.
- *Open questions*:
    - If approximate inference method X will be used at test time, what conditions imply that it is best to train by optimizing log loss with the same approximate inference method X?
    - What type of (approximate) inference is best at test time after pseudolikelihood and composite likelihood training?
    - For relational models (with templated factors), does the relative performance of the different estimators change?

Note: Model size (number of variables) seems to be an unimportant factor in choosing a learning method. The tractability (treewidth) of the model is a better measure of the computational cost; for a fixed treewidth, the cost of inference scales linearly in the size of the model.

Data

- *How many training examples are available?*
    - If the training set is very large and a gradient-based optimization method is being used, then stochastic gradient descent is a good option for CRFs. For intractable MRFs, stochastic gradient is mainly useful if the time required for approximate inference is outweighed by the time required to compute a batch estimate of the gradient (after inference); for this situation to occur, the training set would need to be very large relative to the size of the model.
- *Is the problem fully supervised or weakly supervised?* (By "weakly" supervised, we mean that the data or data statistics are only partially observed. This setting includes models with latent variables, as well as outside knowledge expressed as constraints.)

  - Fully supervised: This thesis uses the supervised setting. Further work is needed to see if composite likelihood or similar ideas may be applied to the weakly supervised setting.
  - Weakly supervised: Some very successful approaches include posterior regularization [K. Ganchev and Taskar, 2010] and generalized expectation [McCallum et al., 2007]. Expectation-maximization [Baum et al., 1970, Dempster et al., 1977] is simple to implement but can be less effective in practice. Posterior regularization and generalized expectation are especially useful since they permit weak supervision via constraints on data statistics, whereas expectation-maximization is designed for the more limited setting with unobserved variable values.

- *Open questions*:
  - Can structured composite likelihood or similar ideas be applied to weakly supervised learning?

Computational Constraints

- *How much computational power is available for training?*
  - Less computational power indicates the need for faster training methods, such as pseudolikelihood and stochastic gradient optimization.
- *Is the computing system distributed?*
  - Decomposable methods such as pseudolikelihood and composite likelihood are especially useful for distributed computing. Other methods for parallelizing parts of learning, such as distributed gradient computation or distributed approximate inference (e.g., Gonzalez et al. [2011]), have been shown to be viable but require much more communication and coordination between compute nodes.

5.3.3 Structure Learning Methods

We list several classes of structure learning methods. Many of these methods use parameter learning (and inference and optimization) as subroutines, so our discussion from Sec. 5.3.1 is relevant here.

- *Chow-Liu and generalizations*: The Chow-Liu algorithm for MRFs [Chow and Liu, 1968] and its generalizations (e.g., Bradley and Guestrin [2010], Shahaf et al. [2009]) are specialized for finding low-treewidth (tractable) models.
- *Local search*: Methods which optimize log loss (or approximations) by making small modifications to the structure (e.g., Teyssier and Koller [2005], Torralba et al. [2004]) are effective in practice but lack strong theoretical guarantees. These methods can also be useful for improving initial structures found by other methods.
- *Sparsifying regularization*: Methods which optimize a loss (such as log loss or pseudolikelihood) and impose sparsity (via, e.g., block-L_1 regularization) can be fast, effective in practice, and accompanied by theoretical guarantees [Ravikumar et al., 2010].
- *Constraint-based methods*: Methods which create a list of constraints via independence tests and then find a structure which fits these constraints can come with theoretical guarantees (e.g., Chechetka and Guestrin [2010]) but are arguably less practical (efficient) than optimization-based methods.

5.3.4 Guide for Practitioners: Structure

Model

- *Is it a CRF or an MRF? Should the model permit tractable inference (have low treewidth)?*
    - Tractable MRF: For trees, use the Chow-Liu algorithm [Chow and Liu, 1968]. For more general models, reasonable existing methods include the method from Shahaf et al. [2009] for learning treewidth-k models and the method from Lowd and Domingos [2008] for learning arithmetic circuits (which may be high treewidth yet tractable).
    - Tractable CRF: We believe our generalized Chow-Liu algorithm using Decomposable Conditional Influence for edge weights is effective (Ch. 3), but much work remains to prove theoretical guarantees and discover alternate methods.
    - Intractable: We recommend block-L_1 regularization-based methods (e.g., Ravikumar et al. [2010], Schmidt et al. [2008]), which are fast, have proven effective in practice, and come with theoretical guarantees [Ravikumar et al., 2010]. Their results may be improved with further steps using local search (e.g., Teyssier and Koller [2005], Torralba et al. [2004]).
- *Open questions*:
    - What types of methods are best when the model class is well-specified vs. misspecified?

Data

- *Open questions*:
    - Are certain learning methods best with small (or large) training sets?
    - How can weak supervision best be used by structure learning methods?

Computational Constraints

- *How much computational power is available for training?*
    - Optimization-based methods such as block-L_1-regularized pseudolikelihood can be very efficient. Local search methods can require much more computation if search converges slowly.
- *Is the computing system distributed?*
    - Decomposable methods such as block-L_1-regularized pseudolikelihood and independence tests are easily distributed. Local search methods may be more difficult to parallelize.

Special Learning Goals

- *Are both parameters and structure being learned?*
    - Constraint-based methods can learn structure without estimating parameters. Optimization-based methods such as local search and block-L_1-regularized pseudolikelihood tend to estimate both jointly.
- *Does the structure need to be interpretable?*
    - If recovering a meaningful (and sparse) structure is important, the method from Liu et al. [2010a] can help with selecting regularization appropriately.

Chapter 6

Future Work

In previous chapters, we discussed more detailed future work on extensions to each of our three topics: CRF parameter learning (Sec. 2.8), CRF structure learning (Sec. 3.6), and parallel regression (Sec. 4.7). In this section, we discuss four broader goals.

We first discuss long-term projects on parameter learning and parallel regression, respectively. In Sec. 6.1, we discuss the goal of unified analyses and broad comparisons of the many parameter learning methods in existence. In Sec. 6.2, we discuss parallel optimization with heterogeneity in both the problems and the computing systems.

We then propose two projects which unify all three topics in this thesis. In Sec. 6.3, we discuss unifying our methods under a common system. Unifying our methods will involve joint analyses of parameter learning, structure learning, and regression, and it has the potential to permit more adaptive learning methods which can better take advantage of data, computation, and parallelism. In Sec. 6.4, we discuss applying this system to the task of machine reading. Such a large real-world problem presents an ideal opportunity for applying our methods, as well as many challenges which inspire new directions for research.

6.1 Unified Analyses of Parameter Learning

In Sec. 2.1.1, we discussed several categories of methods for learning the parameters of probabilistic graphical models. Different authors have given evidence for different methods outperforming others in different settings. Very few empirical or theoretical works have done broad comparisons across either methods or problem settings. Greater efforts both at broad empirical comparisons and at unified analyses would provide valuable guidance for current applications and for future development of improved learning methods.

We find inspiration for this goal in the field of optimization, in which a variety of efforts have coalesced into an increasingly well-defined toolkit. This optimization toolkit includes a set of often-exchangeable parts (batch vs. stochastic gradient, coordinate vs. full-gradient descent, first- vs. second-order methods, etc.) which are known to work well in certain situations (e.g., batch with few training examples and stochastic with many training examples).

Such a toolkit is in very early stages in the field of parameter learning. Some parts which we envision in a toolkit are:

Batch vs. stochastic: Objectives and gradients may be computed using batch or stochastic estimates. A number of works have compared these two methods empirically (e.g., Vishwanathan et al. [2006]), and much analysis from the optimization literature applies to our parameter learning problem.

Optimization vs. matching: Optimization-based learning includes maximum likelihood estimation and related methods which pose learning objectives, such as pseudolikelihood and composite likelihood. Matching-based learning refers to methods such as the method of moments and the canonical parameterization (Sec. 2.6) which do not pose optimization objectives but instead match terms between the model and empirical distributions (moments for the method of moments, and local conditional distributions for the canonical parameterization). Very little work has compared these two categories of learning methods, though some works have discussed combinations [Cam, 1986].

Optimization objectives: Among optimization-based learning, a variety of objectives have been proposed, including those which we discuss (likelihood, pseudolikelihood, and composite likelihood) and others such as the objectives used by contrastive divergence [Hinton, 2002], piecewise likelihood [Sutton and McCallum, 2005], score matching [Hyvärinen, 2005] and ratio matching [Hyvärinen, 2007]. Theoretical and empirical comparisons include Liang and Jordan [2008], Bradley and Guestrin [2012], and Marlin and de Freitas, which examine various subsets of these objectives.

Exact vs. approximate inference: The optimization-based approaches all use some type of inference, either on the full model or estimator components. Though approximate inference has been shown to be empirically successful during learning, it is still unclear how various approximate inference methods interact with and relate to learning methods. One notable exception is the analysis of learning with variational inference by Wainwright [2006], but many other settings lack such theoretical results.

Currently, certain methods tend to be preferred in certain areas, possibly because of tradition within the literature. Broad comparisons of methods might help to break down such boundaries. For example, deep learning researchers often use stochastic gradient for optimization, but recent work suggested that some batch methods may be much faster [Le et al., 2011].

Finally, we mention two major issues which should impact how this learning toolkit is constructed: test-time inference and semi-supervision. Some research has indicated that learning methods should match the type of inference used at test-time [Wainwright, 2006]. In Wainwright [2006], the same type of approximate inference is used for both training and testing, but for many learning methods (e.g., the method of moments), it is less clear how to choose a matching inference method. It is also unclear which ideas from supervised parameter learning translate to the semi-supervised setting; we mention issues with composite likelihood in Sec. 2.8.3. Including test-time inference and semi-supervision in analysis may significantly affect the relative performance of methods in a parameter learning toolkit.

6.2 Parallel Optimization in Heterogeneous Settings

We discussed parallel regression for the multicore setting in Ch. 4, as well as proposed extensions to the distributed setting (Sec. 4.7.4). Even more interesting and important will be extensions to more complex learning problems and computing environments. Many learning problems, such as CRF parameter learning, have complex structure which can make parallelization more difficult to design. Similarly, large-scale parallel computing systems tend to have complex, hierarchical structure, with parallelism at multiple levels: distributed, multi-core (CPU) and many-core (Graphics Processing Units). Designing learning methods for such heterogeneous problems and systems will require adaptively dividing learning into many levels of granularity.

Computing systems for handling granular parallel computation are already being developed. For example, MapReduce [Dean and Ghemawat, 2004] handles data-parallel computation, while the more recent GraphLab [Low et al., 2010] handles more complex graph-structured computation.

Our primary interest, however, is not in system design; rather, it is in adapting and designing the learning methods themselves with parallelism in mind. As we argued throughout this thesis, the choice of learning method affects the potential for parallelism. For parameter learning, composite likelihood estimators may be chosen to permit highly data-parallel computation; for regression, coordinate descent permits data-parallel computation for some problems but not others. Likewise, the method for parallelization affects the efficacy of the learning method. For some composite likelihood estimators, distributed computation may limit communication and force us into the disjoint optimization regime (with higher sample complexity); for `Shotgun`, too much data-parallel computation (too many parallel updates) may slow convergence. Thus, we would like to design learning methods with parallelism in mind, including sample complexity (and accuracy) in our analysis.

It is not clear which method for parallelizing learning is ideal for each type of parallel computation. For example, in distributed computation, it is easy to parallelize by splitting training examples across compute nodes [Zhang et al., 2012, Zinkevich et al., 2010], but splitting features may be better in some settings, even if it requires more communication [Boyd et al., 2011a]. Initially, we hope to address this question in the context of distributed regression, in which we may parallelize over training examples, features, or both, using distributed and multicore parallelism. Ultimately, we would like to address optimizing parallelism for learning graphical models, whose often heterogeneous and hierarchical structure may require much more flexible definitions of granularity for parallelism.

6.3 Unifying Our Methods

Our works on CRF parameter and structure learning and parallel regression have been largely separate, yet these problems are closely linked. We discuss plans for a distributed system combining all three parts (Fig. 6.1). Each part of learning can be used for solving subproblems in other parts of learning. Linking these methods into a single pipeline for learning CRFs will produce a flexible, highly scalable system.

In addition to practical system building, this research effort will give rise to a plethora of questions about interactions between the various components. A joint learning system will have many more knobs for optimizing trade-offs and adapting to new models and data, and each link in the system will require new analyses. We detail major challenges and questions as we outline our planned system in the rest of this section.

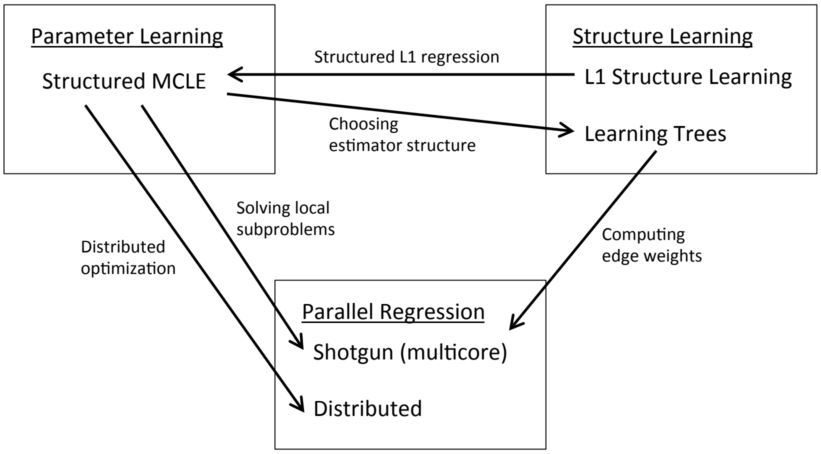


Figure 6.1: **Planned CRF learning system.** This figure shows how our current and future work on CRF parameter and structure learning and parallel regression will be tied together in our planned CRF learning system. Each link in the system presents challenges in analyzing how to integrate parts of learning, as well as opportunities for sharing innovations between those parts.

6.3.1 Parameter Learning

We plan to learn parameters using structured maximum composite likelihood estimation (MCLE) (Ch. 2). We will distribute computation across compute nodes by solving subsets of the estimator components on each node, with limited communication between components.

The primary challenge will be automatically optimizing our choice of composite likelihood estimator structure and its implementation in a distributed system. This optimization will be w.r.t. several objectives (or constraints): minimize sample complexity, minimize computational complexity (of inference in each estimator component), maximize potential for parallelism (across distributed compute nodes), and minimize the amount of communication (between compute nodes). Refer back to Ch. 2 and particularly the future work in Sec. 2.8 for discussions on how estimator structure and distributed optimization with limited communication affect these objectives.

This design will let us take advantage of our work on structure learning and parallel regression. Choosing structures for estimator components may be able to take advantage of our work on learning tree CRFs; one interesting question is how traditional structure learning differs choosing a good estimator structure. On each compute node, optimizing each estimator component is essentially a regression problem, which can use our work on `Shotgun` for multicore regression. Across compute nodes, we would like to use limited communication to improve sample complexity, which may be closely related to our work on distributed regression.

6.3.2 Structure Learning

We plan to learn two types of structures: sparse structures (for general models) and trees (for tractable inference). For general sparse structures, we will generalize L_1-regularized pseudolikelihood to L_1-regularized structured composite likelihood (mentioned in Sec. 3.6.3). For trees, we plan to improve upon our current algorithmic approach discussed in Ch. 3.

A primary challenge in learning sparse structures will be developing a method which adaptively chooses a structured estimator which can outperform unstructured pseudolikelihood estimators. One possibility is to use an incremental approach which iteratively constructs more complex estimators as edges are chosen for the model. Learning trees presents many challenges, outlined in Sec. 3.6.

Our structure learning methods will benefit from our work on parameter learning and parallel regression. Our choice of composite likelihood estimators for L_1-based structure learning will be guided by our discoveries about composite likelihood for the domain of parameter learning. Our tree-learning algorithms will likely use regression as a basic component for computing edge weights, so `Shotgun` will allow immediate speedups from multicore parallelism.

6.3.3 Parallel Regression

We plan to integrate our current work on `Shotgun` for multicore regression (Ch. 4), which will be immediately applicable to both parameter and structure learning. Our planned work on distributed regression (Sec. 4.7.4) will be closely related to our plans for distributed optimization of structure composite likelihood.

One novel challenge for multicore regression will be taking advantage of the fact that we will be solving many related regression problems (e.g., regressing variables on neighbors within the same region of a CRF). The primary challenge for distributed parameter learning will be limiting communication, which may take two forms. First, we may limit the number of messages each compute node may pass to other nodes; ideally, our analysis will prove that a small number of communications can achieve sample complexity close to joint optimization while allowing parallel speedups close to disjoint optimization (mentioned in Sec. 2.8.1). Second, we may communicate sparse data (e.g., sparse parameter vectors), as discussed in Sec. 4.7.4.

6.4 Machine Reading

We discuss plans for applying our unified CRF learning system to machine reading. Our goals for machine reading center around building a system which reads natural language and improves its performance and understanding over time. Our ideas are largely inspired by the *Never-Ending Language Learner (NELL)*, developed by Carlson et al. [2010]. NELL, described more in Sec. 6.4.1, is a system designed to learn forever with minimal human intervention, iteratively constructing a knowledge base from reading online text and improving its reading using its knowledge base.

Our goals (Sec. 6.4.2) center around creating a system like NELL, but phrased as a single probabilistic model such as a CRF. Related ideas for machine reading using probabilistic models include the systems proposed by Wick et al. [2010], who presented probabilistic databases phrased as factor graphs, and by Poon and Domingos [2010], who discussed databases phrased as Markov logic networks [Richardson and Domingos, 2006].

6.4.1 Never-Ending Language Learner (NELL)

NELL is a system for extracting a knowledge base from web pages, designed to iteratively expand its set of beliefs with minimal human intervention. NELL is composed of several components which are designed to extract knowledge from text using orthogonal methods. In each iteration, components independently extract new knowledge; the results from all components are aggregated; and each component uses the aggregated knowledge in the next iteration. NELL, as described in Carlson et al. [2010], has four such components for finding instances of categories and binary relations:

Coupled Pattern Learner: Uses contextual patterns ("mayor of X")

Coupled Set Expander for Any Language: Uses lists and tables on web pages

Coupled Morphological Classifier: Uses text features such as words, capitalization, and parts of speech.

Rule Learner: Uses first-order relational learning, represented as probabilistic Horn clauses.

6.4.2 Our Proposals

A system like NELL represents both an important application which can benefit from our CRF parameter learning methods, and a set of challenges for our methods which will inspire further research. Initially, we plan to phrase components from NELL and other machine reading systems as CRFs. We then plan to link these various components together into a single reading system.

Each of the problems solved by the components of NELL could be phrased as a CRF. To illustrate, we discuss our initial work on the category prediction task for the Coupled Pattern Learner (CPL) component.

Phrased in our notation for CRFs, one example for CPL corresponds to one noun phrase. CPL's input variables $\mathcal{X}$ are counts of occurrences of the noun phrase in each contextual pattern, and each output variable Y_i is a binary indicator of whether the noun phrase belongs to a particular category. Currently, NELL does not solve this problem with a traditional probabilistic model. Carlson [2010] tested supervised logistic regression, as well as a semi-supervised coupled version of logistic regression, and those results indicated that more principled probabilistic models which account for structure in $\mathcal{Y}$ can give better results. We therefore expect our CRF learning methods to give further improvements.

We also plan to test our structure learning methods on NELL. For our tree CRF learning methods, we would test projecting the CPL model structure onto a tree structure, thus trading off efficiency and accuracy. For methods which learn general structures, we would be able to learn new dependencies between categories.

Applying CRFs to NELL subproblems poses several major challenges. We would need to develop extensions of our methods to the semi-supervised setting and to more natural graph structures, as discussed in Sec. 2.8 and Sec. 3.6. The scale of NELL and need for never-ending learning might require more research into prioritizing parts of learning; the decomposability of our methods will likely prove useful in allowing us to focus learning on particular parts of the model. Wick et al. [2010] discuss similar ideas for focused learning.

Linking multiple machine reading systems into a single system will likely require a hierarchical CRF and significant work to calibrate the different subsystems. However, a joint probabilistic model may improve performance over systems such as NELL by modeling correlations between the subsystems' beliefs or predictions.

We believe that the principled approach of probabilistic graphical models and the power and flexibility of our learning methods will let us make several important contributions to the field of machine reading. First, our methods should improve performance, both in accuracy and scalability. A probabilistic graphical model-based approach should produce better estimates of probabilities when extracting knowledge from text, and our learning methods should make such an approach feasible on even larger knowledge bases and data streams. Second, the ability to compute meaningful probabilities should be very helpful for active learning, which will be key in a large, growing system meant to improve with limited human supervision. Finally, the decompositions in our learning methods should fit well with the focused, ongoing learning approaches which work well for machine reading, as in the systems designed by Carlson et al. [2010] and Wick et al. [2010]. For example, focusing learning could take the form of devoting extra computation to optimizing certain components of parameter estimators. Our methods should also be adaptable to ongoing learning, in which model size may increase with new knowledge or data, for our estimators would only need to be updated locally in the modified regions of the model.

Appendix A

CRF Parameter Learning

A.1 Composite Likelihood: Proofs from Ch. 2

A.1.1 CRF Losses and Derivatives

We provide a list of losses and derivatives for general log-linear CRFs.

General log-linear CRF model:

$$\begin{aligned} P_\theta(Y|X) &= \frac{1}{Z(X;\theta)} \exp\left(\theta^T \phi(Y,X)\right) & \text{(A.1)} \\ &= \frac{1}{Z(X;\theta)} \exp\left(\sum_j \theta_j^T \phi_j(Y_{C_j}, X_{D_j})\right), \end{aligned}$$

where $\theta \in \mathbb{R}$ is a length-r vector of parameters and $\phi(Y,X) \in \mathbb{R}_+$ is a length-r non-negative feature vector. When discussing the factors making up the model, we express factor j with domain (Y_{C_j}, X_{D_j}) in terms of its corresponding parameters θ_j and features ϕ_j.

Log loss for model P_θ w.r.t. distribution P_{θ^*}:

$$\begin{aligned} \ell_L(\theta) &= \mathbf{E}_{P(X)}\left[\mathbf{E}_{P_{\theta^*}(Y|X)}\left[-\log P_\theta(Y|X)\right]\right] & \text{(A.2)} \\ &= \mathbf{E}_{P(X)}\left[\mathbf{E}_{P_{\theta^*}(Y|X)}\left[-\theta^T\phi(Y,X)\right] + \log Z(X;\theta)\right]. \end{aligned}$$

Gradient of log loss

$$\nabla \ell_L(\theta) = \mathbf{E}_{P(X)}\left[-\mathbf{E}_{P_{\theta^*}(Y|X)}\left[\phi(Y,X)\right] + \mathbf{E}_{P_\theta(Y'|X)}\left[\phi(Y',X)\right]\right]. \quad \text{(A.3)}$$

Hessian of log loss:

$$\begin{aligned} \nabla^2 \ell_L(\theta) &= \mathbf{E}_{P(X)}\left[\mathbf{E}_{P_\theta(Y'|X)}\left[\phi(Y',X)^{\otimes}\right] - \left(\mathbf{E}_{P_\theta(Y'|X)}\left[\phi(Y',X)\right]\right)^{\otimes}\right] & \text{(A.4)} \\ &= \mathbf{E}_{P(X)}\left[\mathbf{Var}_{P_\theta(Y'|X)}\left[\phi(Y',X)\right]\right] & \text{(A.5)} \end{aligned}$$

Third derivative of log loss:

$$\begin{aligned}&\frac{\partial}{\partial \theta_i} \nabla^2 \ell_L(\theta) \\ &\quad = \mathbf{E}_{P(X)} \left[\mathbf{E}\left[\phi_i \phi^{\otimes}\right] + 2\mathbf{E}\left[\phi_i\right] \mathbf{E}\left[\phi\right]^{\otimes} - \mathbf{E}\left[\phi_i\right] \mathbf{E}\left[\phi^{\otimes}\right] - \mathbf{E}\left[\phi\right] \mathbf{E}\left[\phi_i \phi\right]^T - \mathbf{E}\left[\phi_i \phi\right] \mathbf{E}\left[\phi\right]^T \right],\end{aligned} \tag{A.6}$$

where all unspecified expectations are w.r.t. $P_\theta(Y'|X)$ and $\phi = \phi(Y', X)$.

Composite likelihood loss for model P_θ w.r.t. distribution P_{θ^*}:

$$\begin{aligned}\ell_{CL}(\theta) &= \mathbf{E}_{P(X)} \left[\mathbf{E}_{P_{\theta^*}(Y|X)} \left[-\sum_i \log P_\theta(Y_{A_i} | Y_{\backslash A_i}, X) \right] \right] \\ &= \mathbf{E}_{P(X)} \left[\mathbf{E}_{P_{\theta^*}(Y|X)} \left[\sum_i -\theta_{A_i}^T \phi_{A_i}(Y, X) + \log \sum_{y'_{A_i}} \theta_{A_i}^T \phi_{A_i}(y'_{A_i}, Y_{\backslash A_i}, X) \right] \right].\end{aligned} \tag{A.7}$$

Above, the likelihood components are specified by subsets $Y_{A_i} \subseteq Y$. The parameter and feature subvectors associated with component i are specified as θ_{A_i} and ϕ_{A_i}; here, "associated with" means that at least one $Y_j \in Y_{A_i}$ is an argument of the function ϕ_k for each element k of ϕ_{A_i}.

Gradient of composite likelihood loss:

$$\nabla \ell_{CL}(\theta) = \mathbf{E}_{P(X)} \left[\mathbf{E}_{P_{\theta^*}(Y|X)} \left[\sum_i -\phi_{A_i}(Y, X) + \mathbf{E}_{P_\theta(Y'_{A_i}|Y_{\backslash A_i}, X)} \left[\phi_{A_i}(Y'_{A_i}, Y_{\backslash A_i}, X) \right] \right] \right]. \tag{A.8}$$

Above, we abuse notation in the vector summation: each element in the summation over i is a subvector of the full length-r gradient; these subvectors should be treated as length-r vectors (with zeros in the extra elements) in the summation.

Hessian of composite likelihood loss:

$$\begin{aligned}&\nabla^2 \ell_{CL}(\theta) \\ &= \mathbf{E}_{P(X) P_{\theta^*}(Y|X)} \left[\sum_i \mathbf{E}_{P_\theta(Y'_{A_i}|Y_{\backslash A_i}, X)} \left[\phi_{Y_{A_i}}(Y'_{A_i}, Y_{\backslash A_i}, X)^{\otimes} \right] - \left(\mathbf{E}_{P_\theta(Y'_{A_i}|Y_{\backslash A_i}, X)} \left[\phi_{Y_{A_i}}(Y'_{A_i}, Y_{\backslash A_i}, X) \right] \right)^{\otimes} \right] \\ &= \mathbf{E}_{P(X) P_{\theta^*}(Y|X)} \left[\sum_i \mathbf{Var}_{P_\theta(Y'_{A_i}|Y_{\backslash A_i}, X)} \left[\phi_{Y_{A_i}}(Y'_{A_i}, Y_{\backslash A_i}, X) \right] \right].\end{aligned} \tag{A.9}$$

In the matrix summation above, we again abuse notation: each element in the sum over i is added to a submatrix of the full $r \times r$ Hessian. Each of these elements is the Hessian for a likelihood component; we denote the Hessian of the i^{th} component as $\nabla^2[\ell_{CL}(\theta)]_{A_i}$.

Third derivative of composite likelihood loss:

$$\begin{aligned}&\frac{\partial}{\partial \theta_t} \nabla^2 \ell_{CL}(\theta) \\ &\quad = \mathbf{E}_{P(X) P_{\theta^*}(Y|X)} \Bigg[\sum_{i \,:\, \theta_t \in \theta_{A_i}} \mathbf{E}\left[\phi_t \phi_{A_i}^{\otimes}\right] + 2\mathbf{E}\left[\phi_t\right] \mathbf{E}\left[\phi_{A_i}\right]^{\otimes} - \mathbf{E}\left[\phi_t\right] \mathbf{E}\left[\phi_{A_i}^{\otimes}\right] \\ &\qquad\qquad - \mathbf{E}\left[\phi_{A_i}\right] \mathbf{E}\left[\phi_t \phi_{A_i}\right]^T - \mathbf{E}\left[\phi_t \phi_{A_i}\right] \mathbf{E}\left[\phi_{A_i}\right]^T \Bigg],\end{aligned} \tag{A.10}$$

where $\theta_t \in \theta_{A_i}$ indicates whether θ_t is an element of the parameter subvector θ_{A_i}. All unspecified expectations are w.r.t. $P_\theta(Y'_{A_i}|Y_{\backslash A_i}, X)$, and the feature function arguments are hidden: $\phi_{A_i} = \phi_{A_i}(Y'_{A_i}, Y_{\backslash A_i}, X)$ and $\phi_t = \phi_t(Y'_{A_i}, Y_{\backslash A_i}, X)$.

A.1.2 Parameter Estimation with MLE

We prove finite sample bounds for regression problems using log-linear models $P(Y|X)$, as defined in Eq. (A.1). This analysis applies both to learning CRF parameters via MLE and to learning parameters for each composite likelihood component (when using disjoint optimization). Our analysis in this section extends the analysis of Ravikumar et al. (2010) for Ising MRFs to the more general setting of log-linear CRFs. Also, while they were concerned with L_1-regularized regression in the high-dimensional setting (with the number of covariates increasing with the training set size), we limit our discussion to L_1- and L_2-regularized regression with a fixed number of covariates.

The log loss and its derivatives are defined in Eq. (A.2), Eq. (A.3), Eq. (A.4), and Eq. (A.6). We train our model by minimizing the regularized log loss over n training samples, as in Eq. (2.2).

All of the results in this subsection are presented in terms of the parameters θ and corresponding features ϕ, each of which are assumed to be length-r vectors. If the feature space is overcomplete, then the constant C_{min}, the minimum eigenvalue of the Hessian of the log loss, will be zero, violating our assumption that $C_{min} > 0$. However, in all of these results, θ and ϕ (and quantities defined in terms of these vectors) may be replaced with $U^T\theta$ and $U^T\phi$, where U is a $r \times d$ matrix whose columns are eigenvectors of the Hessian corresponding to the non-zero eigenvalues. This transformation using U projects the parameters and features onto a minimal set of vectors spanning the feature space.

The following lemma (similar to Lemma 2 of Ravikumar et al. (2010)) lets us bound the max norm of the gradient of the empirical log loss at θ^* with high probability.

Lemma A.1.1. *Given n samples, a bound ϕ_{max} on the magnitude of each of r features, and a constant $\delta > 0$, we have*

$$P\left[\|\nabla\hat{\ell}_L(\theta^*)\|_\infty > \delta\right] \leq 2r \exp\left(-\frac{\delta^2 n}{2\phi_{max}^2}\right). \tag{A.11}$$

Proof of Lemma A.1.1: Consider one element j of the gradient at θ^*: $[\nabla\hat{\ell}_L(\theta^*)]_j$, as in Eq. (A.3). This element is a random variable (random w.r.t. the sample) with mean 0. The variable also has bounded range $[-\phi_{max}, \phi_{max}]$. We may therefore apply the Azuma-Hoeffding inequality Hoeffding [1963], which states that

$$P\left[\left|[\nabla\hat{\ell}_L(\theta^*)]_j\right| > \delta\right] \leq 2\exp\left(-\tfrac{\delta^2 n}{2\phi_{max}^2}\right). \tag{A.12}$$

Applying a union bound over all r elements of the gradient gives the lemma's result. ■

We now give a lemma which shows that the minimum eigenvalue of the Hessian of the empirical log loss $\hat{\ell}_L$ (w.r.t. n samples from the target distribution) is close to that for the actual log loss ℓ_L (w.r.t. the target distribution itself).

Lemma A.1.2. *Assume $\Lambda_{min}\left(\nabla^2\ell_L(\theta^*)\right) \geq C_{min} > 0$ and $\phi_{max} = \max_{j,y,x}\phi_j(y,x)$. With n training samples, the minimum eigenvalue of the Hessian of the empirical log loss is not much smaller than C_{min}*

with high probability:

$$P\left[\Lambda_{min}\left(\nabla^2\hat{\ell}_L(\theta^*)\right) \leq C_{min} - \epsilon\right] \leq 2r^2 \exp\left(-\frac{n\epsilon^2}{8r^2\phi_{max}^4}\right). \tag{A.13}$$

Proof of Lemma A.1.2: Our proof is similar to that of Lemma 5 from Ravikumar et al. (2010). Define shorthand for the Hessian of the log loss w.r.t. the target distribution: $\mathcal{Q} \doteq \nabla^2\ell_L(\theta^*)$ and for the Hessian w.r.t. the empirical distribution: $\mathcal{Q}^n \doteq \nabla^2\hat{\ell}_L(\theta^*)$. Using the Courant-Fischer variational representation Horn and Johnson [1990], we can re-express the minimum eigenvalue of the Hessian:

$$\begin{aligned}
\Lambda_{min}(\mathcal{Q}) &= \min_{\|v\|_2=1} v^T\mathcal{Q}v && \text{(A.14)}\\
&= \min_{\|v\|_2=1} \left[v^T\mathcal{Q}^n v + v^T(\mathcal{Q}-\mathcal{Q}^n)v\right] && \text{(A.15)}\\
&\leq v_{min}^T\mathcal{Q}^n v_{min} + v_{min}^T(\mathcal{Q}-\mathcal{Q}^n)v_{min}, && \text{(A.16)}\\
& && \text{(A.17)}
\end{aligned}$$

where $v_{min} : \|v_{min}\|_2 = 1$ is an eigenvector corresponding to the minimum eigenvalue of $\mathcal{Q}^n$. Rearranging,

$$\begin{aligned}
\Lambda_{min}(\mathcal{Q}^n) &\geq \Lambda_{min}(\mathcal{Q}) - v_{min}^T(\mathcal{Q}-\mathcal{Q}^n)v_{min} && \text{(A.18)}\\
&\geq \Lambda_{min}(\mathcal{Q}) - \Lambda_{max}(\mathcal{Q}-\mathcal{Q}^n) && \text{(A.19)}\\
&\geq \Lambda_{min}(\mathcal{Q}) - \left(\sum_{s=1}^{r}\sum_{t=1}^{r}(\mathcal{Q}_{st}-\mathcal{Q}_{st}^n)^2\right)^{1/2}, && \text{(A.20)}
\end{aligned}$$

where the last inequality upper-bounded the spectral norm with the Frobenius norm. Recall that

$$\nabla^2\hat{\ell}_L(\theta^*) = \tfrac{1}{n}\sum_{i=1}^{n}\mathbf{E}_{P_{\theta^*}(Y'|x^{(i)})}\left[\phi(Y',x^{(i)})^{\otimes}\right] - \left(\mathbf{E}_{P_{\theta^*}(Y'|x^{(i)})}\left[\phi(Y',x^{(i)})\right]\right)^{\otimes} \tag{A.21}$$

We upper-bound the Frobenius norm term by noting that each element $(\mathcal{Q}_{st} - \mathcal{Q}_{st}^n)$ may be written as an expectation over our n samples of zero-mean, bounded-range values. Abbreviating $\phi = \phi(Y,X)$ and $\phi^{(i)} = \phi(Y,x^{(i)})$, we can write:

$$\begin{aligned}
\mathcal{Q}_{st} - \mathcal{Q}_{st}^n &= \mathbf{E}_{P(X)}\left[\mathbf{E}_{P_{\theta^*}(Y|X)}\left[\phi_s\phi_t\right] - \mathbf{E}_{P_{\theta^*}(Y|X)}\left[\phi_s\right]\mathbf{E}_{P_{\theta^*}(Y|X)}\left[\phi_t\right]\right] && \text{(A.22)}\\
&\quad -\tfrac{1}{n}\sum_{i=1}^{n}\left[\mathbf{E}_{P_{\theta^*}(Y|x^{(i)})}\left[\phi_s^{(i)}\phi_t^{(i)}\right] - \mathbf{E}_{P_{\theta^*}(Y|x^{(i)})}\left[\phi_s^{(i)}\right]\mathbf{E}_{P_{\theta^*}(Y|x^{(i)})}\left[\phi_t^{(i)}\right]\right] && \text{(A.23)}\\
&= \tfrac{1}{n}\sum_{i=1}^{n}\Bigg[\mathbf{E}_{P(X)}\left[\mathbf{E}_{P_{\theta^*}(Y|X)}\left[\phi_s\phi_t\right] - \mathbf{E}_{P_{\theta^*}(Y|X)}\left[\phi_s\right]\mathbf{E}_{P_{\theta^*}(Y|X)}\left[\phi_t\right]\right] && \text{(A.24)}\\
&\qquad - \left[\mathbf{E}_{P_{\theta^*}(Y|x^{(i)})}\left[\phi_s^{(i)}\phi_t^{(i)}\right] - \mathbf{E}_{P_{\theta^*}(Y|x^{(i)})}\left[\phi_s^{(i)}\right]\mathbf{E}_{P_{\theta^*}(Y|x^{(i)})}\left[\phi_t^{(i)}\right]\right]\Bigg] && \text{(A.25)}
\end{aligned}$$

Each of these n values has magnitude at most $2\phi_{max}^2$. The Azuma-Hoeffding inequality Hoeffding [1963] tells us that, for any s, t,

$$P\left[(\mathcal{Q}_{st}-\mathcal{Q}_{st}^n)^2 \geq \epsilon^2\right] = P\left[|\mathcal{Q}_{st}-\mathcal{Q}_{st}^n)| \geq \epsilon\right] \leq 2\exp\left(-\frac{n\epsilon^2}{8\phi_{max}^4}\right). \tag{A.26}$$

A union bound over all elements s, t shows:

$$P\left[\sum_{s=1}^{r}\sum_{t=1}^{r}(\mathcal{Q}_{st} - \mathcal{Q}^n_{st})^2 \geq r^2\epsilon^2\right] \;\leq\; 2r^2 \exp\left(-\frac{n\epsilon^2}{8\phi^4_{max}}\right) \tag{A.27}$$

$$P\left[\left(\sum_{s=1}^{r}\sum_{t=1}^{r}(\mathcal{Q}_{st} - \mathcal{Q}^n_{st})^2\right)^{1/2} \geq \epsilon\right] \;\leq\; 2r^2 \exp\left(-\frac{n\epsilon^2}{8r^2\phi^4_{max}}\right). \tag{A.28}$$

Using the above inequality with Eq. (A.20), we get:

$$P\left[\Lambda_{min}(\mathcal{Q}^n) \leq C_{min} - \epsilon\right] \;\leq\; 2r^2 \exp\left(-\frac{n\epsilon^2}{8r^2\phi^4_{max}}\right). \quad \blacksquare \tag{A.29}$$

We next prove a lemma which lower-bounds the log loss (w.r.t. our training samples) in terms of the parameter estimation error (the distance between our estimated parameters $\hat{\theta}$ and the target parameters θ^*). The analysis resembles part of Lemma 3 from Ravikumar et al. (2010).

Lemma A.1.3. *Let $\hat{\ell}_L(\theta)$ be the log loss w.r.t. n training samples. Assume bounds $\Lambda_{min}\left(\nabla^2 \ell_L(\theta^*)\right) \geq C_{min} > 0$ and $\phi_{max} = \max_{j,y,x} \phi_j(y, x)$. Let $\delta > 0$. Let $B = \|\theta - \theta^*\|_1$. Then*

$$\hat{\ell}_L(\theta) - \hat{\ell}_L(\theta^*) \geq -\delta B + \tfrac{r^{-1}}{4} C_{min} B^2 - \tfrac{1}{2}\phi^3_{max} B^3 \tag{A.30}$$

with probability at least

$$1 - 2r \exp\left(-\tfrac{\delta^2 n}{2\phi^2_{max}}\right) - 2r^2 \exp\left(-\tfrac{nC^2_{min}}{2^5 r^2 \phi^4_{max}}\right). \tag{A.31}$$

Proof of Lemma A.1.3: Let $u = \theta - \theta^*$ and $\|u\|_1 = B$. Use a Taylor expansion of $\hat{\ell}_L$ around θ^*:

$$\hat{\ell}_L(\theta) \;=\; \hat{\ell}_L(\theta^*) + \left(\nabla \hat{\ell}_L(\theta^*)\right)^T u + \tfrac{1}{2} u^T \left(\nabla^2 \hat{\ell}_L(\theta^*)\right) u + \tfrac{1}{6}\sum_i u_i u^T \left(\tfrac{\partial}{\partial \theta_i} \nabla^2 \hat{\ell}_L(\bar{\theta})\big|_{\bar{\theta}=\theta^*+\alpha u}\right) u \tag{A.32}$$

where $\alpha \in [0, 1]$. We now lower-bound the first-, second-, and third-order terms.

First-order term in Eq. (A.32):

$$\left(\nabla \hat{\ell}_L(\theta^*)\right)^T u \;\geq\; -\left|\left(\nabla \hat{\ell}_L(\theta^*)\right)^T u\right| \tag{A.33}$$

$$\geq\; -\|\nabla \hat{\ell}_L(\theta^*)\|_\infty \|u\|_1 \tag{A.34}$$

$$=\; -\|\nabla \hat{\ell}_L(\theta^*)\|_\infty B \tag{A.35}$$

$$\geq\; -\delta B, \tag{A.36}$$

where the second inequality uses Holder's inequality, and where the last inequality uses Lemma A.1.1 and holds with probability at least $1 - 2r\exp(-\frac{\delta^2 n}{2\phi^2_{max}})$.

Second-order term in Eq. (A.32):

$$\tfrac{1}{2} u^T \left(\nabla^2 \hat{\ell}_L(\theta^*)\right) u \;\geq\; \tfrac{1}{2}\Lambda_{min}\left(\nabla^2 \hat{\ell}_L(\theta^*)\right) \|u\|_2^2 \tag{A.37}$$

$$\geq\; \tfrac{r^{-1}}{2}\Lambda_{min}\left(\nabla^2 \hat{\ell}_L(\theta^*)\right) \|u\|_1^2 \tag{A.38}$$

$$=\; \tfrac{r^{-1}}{2}\Lambda_{min}\left(\nabla^2 \hat{\ell}_L(\theta^*)\right) B^2, \tag{A.39}$$

where we used the definition of $\Lambda_{min}(\nabla^2\hat{\ell}_L(\theta^*))$, the minimum eigenvalue of the Hessian at θ^*. Now use Lemma A.1.2 with $\epsilon = \frac{C_{min}}{2}$ to show:

$$\tfrac{1}{2}u^T\left(\nabla^2\hat{\ell}_L(\theta^*)\right)u \quad \geq \quad \tfrac{r^{-1}}{4}C_{min}B^2, \tag{A.40}$$

which holds with probability at least $1-2r^2\exp\left(-\frac{nC_{min}^2}{2^5r^2\phi_{max}^4}\right)$.

Third-order term in Eq. (A.32):

$$\begin{aligned}
&\tfrac{1}{6}\sum_i u_i u^T\left(\tfrac{\partial}{\partial\bar{\theta}_i}\nabla^2\hat{\ell}_L(\bar{\theta})|_{\bar{\theta}=\theta^*+\alpha u}\right)u && \text{(A.41)}\\
&= \tfrac{1}{6}\sum_i u_i u^T\Big(\mathbf{E}\left[\phi_i\phi^{\otimes}\right] + 2\mathbf{E}\left[\phi_i\right]\mathbf{E}\left[\phi\right]^{\otimes} - \mathbf{E}\left[\phi_i\right]\mathbf{E}\left[\phi^{\otimes}\right] && \text{(A.42)}\\
&\qquad\qquad -\mathbf{E}\left[\phi\right]\mathbf{E}\left[\phi_i\phi\right]^T - \mathbf{E}\left[\phi_i\phi\right]\mathbf{E}\left[\phi\right]^T\Big)u,
\end{aligned}$$

where all expectations are w.r.t. $P_{\theta^*+\alpha u}(Y'|X)$ and $\phi = \phi(Y', X)$. Continuing,

$$\begin{aligned}
&\tfrac{1}{6}\sum_i u_i u^T\Big(\mathbf{E}\left[\phi_i\phi^{\otimes}\right] + 2\mathbf{E}\left[\phi_i\right]\mathbf{E}\left[\phi\right]^{\otimes} - \mathbf{E}\left[\phi_i\right]\mathbf{E}\left[\phi^{\otimes}\right] && \text{(A.43)}\\
&\qquad\qquad -\mathbf{E}\left[\phi\right]\mathbf{E}\left[\phi_i\phi\right]^T - \mathbf{E}\left[\phi_i\phi\right]\mathbf{E}\left[\phi\right]^T\Big)u\\
&= \tfrac{1}{6}\sum_i u_i\Big(\mathbf{E}\left[\phi_i(u^T\phi)^2\right] + 2\mathbf{E}\left[\phi_i\right]\mathbf{E}\left[u^T\phi\right]^2 - \mathbf{E}\left[\phi_i\right]\mathbf{E}\left[(u^T\phi)^2\right] && \text{(A.44)}\\
&\qquad\qquad -2\mathbf{E}\left[u^T\phi\right]\mathbf{E}\left[\phi_i u^T\phi\right]\Big)\\
&= \tfrac{1}{6}\Big(\mathbf{E}\left[(u^T\phi)^3\right] + 2\mathbf{E}\left[u^T\phi\right]^3 - 3\mathbf{E}\left[u^T\phi\right]\mathbf{E}\left[(u^T\phi)^2\right]\Big) && \text{(A.45)}\\
&\geq \tfrac{1}{6}\Big(3\mathbf{E}\left[u^T\phi\right]^3 - 3\mathbf{E}\left[u^T\phi\right]\mathbf{E}\left[(u^T\phi)^2\right]\Big) && \text{(A.46)}\\
&= -\tfrac{1}{2}\mathbf{E}\left[u^T\phi\right]\Big(\mathbf{E}\left[(u^T\phi)^2\right] - \mathbf{E}\left[u^T\phi\right]^2\Big) && \text{(A.47)}\\
&\geq -\tfrac{1}{2}\left|\mathbf{E}\left[u^T\phi\right]\right|\cdot\Big|\mathbf{E}\left[(u^T\phi)^2\right] - \mathbf{E}\left[u^T\phi\right]^2\Big| && \text{(A.48)}\\
&\geq -\tfrac{1}{2}\left|\mathbf{E}\left[u^T\phi\right]\right|\cdot\mathbf{E}\left[(u^T\phi)^2\right] && \text{(A.49)}\\
&\geq -\tfrac{1}{2}\mathbf{E}\left[|u^T\phi|\right]\cdot\mathbf{E}\left[|u^T\phi|^2\right] && \text{(A.50)}\\
&\geq -\tfrac{1}{2}\|u\|_1^3\phi_{max}^3 && \text{(A.51)}\\
&= -\tfrac{1}{2}B^3\phi_{max}^3. && \text{(A.52)}
\end{aligned}$$

Two of our bounds in this proof had small probabilities of failure. Using a union bound, we get the probability of at least one failing, finishing the proof. ■

We now prove a lemma which bounds our parameter estimation error in terms of our training sample size; it is similar to Lemma 3 from Ravikumar et al. (2010). Note that $\hat{\theta}$ is defined as the minimizer of Eq. (2.2) with $\hat{\ell} = \hat{\ell}_L$.

Proof of Theorem 2.3.1: Define a convex function $G : \mathbb{R}^r \to \mathbb{R}$ by

$$G(u) = \hat{\ell}_L(\theta^* + u) - \hat{\ell}_L(\theta^*) + \lambda\left(\|\theta^* + u\|_p - \|\theta^*\|_p\right). \tag{A.53}$$

By definition of $\hat{\theta}$, the function G is minimized at $\hat{u} = \hat{\theta} - \theta^*$. Since $G(0) = 0$, we know $G(\hat{u}) \leq 0$. Using the same argument as Ravikumar et al. (2010), if $G(u) > 0$ for all $u \in \mathbb{R}^r$ with $\|u\|_1 = B$ for some $B > 0$, then we know that $\|\hat{u}\|_1 \leq B$.

Let $u \in \mathbb{R}^r$ with $\|u\|_1 = B$. Using Lemma A.1.3, we can lower-bound G:

$$\begin{aligned} G(u) \quad &\geq \quad -\delta B + \tfrac{r^{-1}}{4} C_{min} B^2 - \tfrac{1}{2}\phi_{max}^3 B^3 \\ &\qquad +\lambda \left(\|\theta^* + u\|_p - \|\theta^*\|_p \right), \end{aligned} \tag{A.54}$$

which holds with the probability given in Lemma A.1.3. We can lower-bound the regularization term (for both L_1 and L_2 regularization):

$$\lambda \left(\|\theta^* + u\|_p - \|\theta^*\|_p \right) \quad \geq \quad -\lambda \|u\|_p \tag{A.55}$$

$$(\text{If } p = 1) \qquad = -\lambda \|u\|_1 = -\lambda B. \tag{A.56}$$

$$(\text{If } p = 2) \qquad = -\lambda \|u\|_2 \geq -\lambda \|u\|_1 = -\lambda B. \tag{A.57}$$

Combine the above bound into Eq. (A.54):

$$G(u) \quad \geq \quad -\delta B + \tfrac{r^{-1}}{4} C_{min} B^2 - \tfrac{1}{2}\phi_{max}^3 B^3 - \lambda B \tag{A.58}$$

$$= \quad B \left[-\delta + \tfrac{r^{-1}}{4} C_{min} B - \tfrac{1}{2}\phi_{max}^3 B^2 - \lambda \right]. \tag{A.59}$$

Note that we need $\lambda > 0, B > 0, \delta > 0$. Eq. (A.59) will be strictly greater than 0 if $B > 0$ and

$$\lambda \quad < \quad -\delta + \tfrac{r^{-1}}{4} C_{min} B - \tfrac{1}{2}\phi_{max}^3 B^2. \tag{A.60}$$

Maximizing this bound w.r.t. B gives $B = \frac{C_{min}}{4r\phi_{max}^3}$. However, we would like for B to shrink as $n^{-1/2}$, the asymptotic rate of convergence, so instead let

$$B = \frac{C_{min}}{4r\phi_{max}^3} n^{-\xi/2}, \tag{A.61}$$

where $\xi \in (0, 1)$. Plugging in this value for B gives

$$\lambda \quad < \quad -\delta + \frac{C_{min}^2}{2^4 r^2 \phi_{max}^3} n^{-\xi/2} - \frac{C_{min}^2}{2^5 r^2 \phi_{max}^3} n^{-\xi}. \tag{A.62}$$

We want to choose δ to be large but still keep $\lambda > 0$, so choose

$$\lambda = \delta = \frac{C_{min}^2}{2^6 r^2 \phi_{max}^3} n^{-\xi/2}, \tag{A.63}$$

which makes Eq. (A.62) hold if $n > 1$. Now that we have chosen δ, we can simplify the probability of failure from Lemma A.1.3:

$$2r \exp\left(-\tfrac{\delta^2 n}{2\phi_{max}^2} \right) + 2r^2 \exp\left(-\tfrac{n C_{min}^2}{2^5 r^2 \phi_{max}^4} \right) \tag{A.64}$$

$$= 2r \exp\left(-\tfrac{C_{min}^4}{2^{13} r^4 \phi_{max}^8} n^{1-\xi} \right) + 2r^2 \exp\left(-\tfrac{C_{min}^2}{2^5 r^2 \phi_{max}^4} n \right) \tag{A.65}$$

$$\leq 2r \exp\left(-\tfrac{C_{min}^4}{2^{13} r^4 \phi_{max}^8} n^{1-\xi} \right) + 2r^2 \exp\left(-\tfrac{C_{min}^4}{2^{13} r^4 \phi_{max}^8} n^{1-\xi} \right) \tag{A.66}$$

$$= 2r(r+1) \exp\left(-\tfrac{C_{min}^4}{2^{13} r^4 \phi_{max}^8} n^{1-\xi} \right). \tag{A.67}$$

Above, we upper-bounded the spectral norm with the Frobenius norm to show that $C_{min} \leq \Lambda_{max}\left(\nabla^2 \hat{\ell}_L(\theta^*)\right) \leq \phi_{max}^2 r$. ■

We can convert the previous result into a sample complexity bound for achieving a given parameter estimation error.

Proof of Corollary 2.3.2: If we wish to have a probability of failure of at most δ when we have n samples, we may choose ξ accordingly:

$$\begin{aligned}
2r(r+1)\exp\left(-\tfrac{C_{min}^4}{2^{13}r^4\phi_{max}^8}n^{1-\xi}\right) &\leq \delta && \text{(A.68)}\\
\log(2r(r+1)) - \tfrac{C_{min}^4}{2^{13}r^4\phi_{max}^8}n^{1-\xi} &\leq \log\delta && \text{(A.69)}\\
\tfrac{C_{min}^4}{2^{13}r^4\phi_{max}^8}n^{1-\xi} &\geq \log\tfrac{2r(r+1)}{\delta} && \text{(A.70)}\\
n^{1-\xi} &\geq \tfrac{2^{13}r^4\phi_{max}^8}{C_{min}^4}\log\tfrac{2r(r+1)}{\delta} && \text{(A.71)}\\
1-\xi &\geq \tfrac{1}{\log n}\left(\log\tfrac{2^{13}r^4\phi_{max}^8}{C_{min}^4} + \log\log\tfrac{2r(r+1)}{\delta}\right) && \text{(A.72)}\\
\xi &\leq 1 - \tfrac{1}{\log n}\left(\log\tfrac{2^{13}r^4\phi_{max}^8}{C_{min}^4} + \log\log\tfrac{2r(r+1)}{\delta}\right). && \text{(A.73)}
\end{aligned}$$

We will set ξ equal to this upper bound in the next part. Likewise, if we wish to have parameter estimation error at most ϵ, then we need:

$$\begin{aligned}
\tfrac{C_{min}}{4r\phi_{max}^3}n^{-\xi/2} &\leq \epsilon && \text{(A.74)}\\
\log\tfrac{C_{min}}{4r\phi_{max}^3} - \tfrac{\xi}{2}\log n &\leq \log\epsilon && \text{(A.75)}\\
\tfrac{\xi}{2}\log n &\geq \log\tfrac{C_{min}}{4r\phi_{max}^3} - \log\epsilon && \text{(A.76)}\\
\tfrac{1}{2}\left(1 - \tfrac{1}{\log n}\left(\log\tfrac{2^{13}r^4\phi_{max}^8}{C_{min}^4} + \log\log\tfrac{2r(r+1)}{\delta}\right)\right)\log n &\geq \log\tfrac{C_{min}}{4r\phi_{max}^3} - \log\epsilon && \text{(A.77)}\\
\log n - \left(\log\tfrac{2^{13}r^4\phi_{max}^8}{C_{min}^4} + \log\log\tfrac{2r(r+1)}{\delta}\right) &\geq 2\log\tfrac{C_{min}}{4r\phi_{max}^3\epsilon}. && \text{(A.78)}
\end{aligned}$$

$$\begin{aligned}
\log n &\geq 2\log\tfrac{C_{min}}{4r\phi_{max}^3\epsilon} + \log\tfrac{2^{13}r^4\phi_{max}^8}{C_{min}^4} + \log\log\tfrac{2r(r+1)}{\delta} && \text{(A.79)}\\
&= \log\tfrac{C_{min}^2}{2^4r^2\phi_{max}^6} + \log\tfrac{2^{13}r^4\phi_{max}^8}{C_{min}^4} + 2\log\tfrac{1}{\epsilon} + \log\log\tfrac{2r(r+1)}{\delta} && \text{(A.80)}\\
&= \log\tfrac{2^9r^2\phi_{max}^2}{C_{min}^2} + 2\log\tfrac{1}{\epsilon} + \log\log\tfrac{2r(r+1)}{\delta} && \text{(A.81)}\\
n &\geq \tfrac{2^9r^2\phi_{max}^2}{C_{min}^2}\tfrac{1}{\epsilon^2}\log\tfrac{2r(r+1)}{\delta}. \quad ■ && \text{(A.82)}
\end{aligned}$$

A.1.3 Parameter Estimation with MCLE

We train our model by minimizing the regularized composite likelihood loss over n training samples:

$$\min_\theta \hat{\ell}_{CL}(\theta) + \lambda\|\theta\|_p, \qquad \text{(A.83)}$$

where $\hat{\ell}_{CL}(\theta)$ is the composite likelihood loss w.r.t. the n training samples, $\lambda \geq 0$ is a regularization parameter, and $p \in \{1,2\}$ specifies the L_1 or L_2 norm. Note that $\hat{\ell}_{CL}(\theta)$ is Eq. (A.7) with $P(X)P_{\theta^*}(Y|X)$ replaced with the empirical distribution.

Lemma A.1.4. *Given n samples, a bound ϕ_{max} on the magnitude of each of r features, a bound M_{max} on the number of likelihood components in which any feature participates, and a constant $\delta > 0$, we have*

$$P\left[\|\nabla \ell_{CL}(\theta^*)\|_\infty > \delta\right] \leq 2r \exp\left(-\frac{\delta^2 n}{2M_{max}^2 \phi_{max}^2}\right). \tag{A.84}$$

Proof of Lemma A.1.4: Consider one element j of the expected gradient at θ^*: $\mathbf{E}_{P(X)P_{\theta^*}(Y|X)}\left[[\nabla \ell_{CL}(\theta^*)]_j\right]$. This element is a random variable (random w.r.t. the sample) with mean 0. The variable also has bounded range $[-M_j\phi_{max}, M_j\phi_{max}]$, where M_j is the number of likelihood components in which θ_j participates. We may therefore apply the Azuma-Hoeffding inequality Hoeffding [1963], which states that

$$P\left[|[\nabla \ell_{CL}(\theta^*)]_j| > \delta\right] \leq 2\exp\left(-\tfrac{\delta^2 n}{2M_j^2 \phi_{max}^2}\right). \tag{A.85}$$

Applying a union bound over all r elements of the gradient and using $M_{max} = \max_j M_j$ gives the lemma's result. ■

Lemma A.1.5. *Define $\rho_t = \sum_{i\,:\,\theta_t \in \theta_{A_i}} \Lambda_{min}(\nabla^2[\ell_{CL}(\theta^*)]_{A_i})$, i.e., the sum of minimum eigenvalues for all likelihood components in which parameter θ_t participates (w.r.t. the target distribution). Define $\hat{\rho}_t$ analogously w.r.t. $\hat{\ell}_{CL}$ (i.e., w.r.t. the empirical distribution). Assume $\phi_{max} = \max_{j,y,x} \phi_j(y,x)$. Let $\mathcal{A}$ denote the set of likelihood components. With n training samples, the empirical quantities $\hat{\rho}_t$ are not much smaller than ρ_t with high probability:*

$$P\left[\exists t \,:\, \hat{\rho}_t \leq \tfrac{\rho_t}{2}\right] \leq 2|\mathcal{A}|r^2 \exp\left(-\frac{nC_{min}^2}{2^5 r^2 \phi_{max}^4}\right). \tag{A.86}$$

Proof of Lemma A.1.5: Let C_i be the minimum eigenvalue of the Hessian of likelihood component A_i w.r.t the target distribution. The Hessian of each likelihood component may be analyzed using Lemma A.1.2, with $\epsilon = \frac{C_i}{2}$, giving the result:

$$P\left[\Lambda_{min}\left(\nabla^2[\hat{\ell}_{CL}(\theta^*)]_{A_i}\right) \leq \tfrac{C_i}{2}\right] \leq 2r_{A_i}^2 \exp\left(-\frac{nC_i^2}{2^5 r_{A_i}^2 \phi_{max}^4}\right), \tag{A.87}$$

where r_{A_i} denotes the length of the parameter vector corresponding to likelihood component A_i. A union bound over all $|\mathcal{A}|$ likelihood components in the min operation gives:

$$P\left[\exists i \,:\, \Lambda_{min}\left(\nabla^2[\hat{\ell}_{CL}(\theta^*)]_{A_i}\right) \leq \tfrac{C_i}{2}\right] \leq 2|\mathcal{A}|r^2 \exp\left(-\frac{nC_{min}^2}{2^5 r^2 \phi_{max}^4}\right). \tag{A.88}$$

I.e., all components' minimum eigenvalues (w.r.t. the training data) are within a factor of $\frac{1}{2}$ of their true eigenvalues (w.r.t. the target distribution) with high probability, which implies that sums of sets of eigenvalues are likewise estimated within a factor of $\frac{1}{2}$, giving the lemma's result. ■

Lemma A.1.6. *Let $\hat{\ell}_{CL}(\theta)$ be the composite likelihood loss w.r.t. n training samples. Assume bounds $\min_i \Lambda_{min}\left(\nabla^2[\ell_{CL}(\theta^*)]_{A_i}\right) \geq C_{min} > 0$ and $\phi_{max} = \max_{j,y,x} \phi_j(y,x)$. Let $\rho_{min} = \min_t \rho_t$. Let M_{max}, M_{min} be the maximum and minimum numbers of components any feature participates in, respectively. Let $\delta > 0$. Let $B = \|\theta - \theta^*\|_1$. Then*

$$\hat{\ell}_{CL}(\theta) - \hat{\ell}_{CL}(\theta^*) \geq -\delta B + \tfrac{r^{-1}}{4}\rho_{min}B^2 - \tfrac{1}{2}M_{max}\phi_{max}^3 B^3 \tag{A.89}$$

with probability at least

$$1 - 2r\exp\left(-\tfrac{\delta^2 n}{2M_{max}^2 \phi_{max}^2}\right) - 2|\mathcal{A}|r^2 \exp\left(-\tfrac{nC_{min}^2}{2^5 r^2 \phi_{max}^4}\right). \tag{A.90}$$

Proof of Lemma A.1.6: We abbreviate this proof where it is similar to that of Lemma A.1.3.

Let $u = \theta - \theta^*$ and $\|u\|_1 = B$. Use a Taylor expansion of $\hat{\ell}_{CL}$ around θ^*:

$$\hat{\ell}_{CL}(\theta) \;=\; \hat{\ell}_{CL}(\theta^*) + \left(\nabla\hat{\ell}_{CL}(\theta^*)\right)^T u + \tfrac{1}{2}u^T\left(\nabla^2\hat{\ell}_{CL}(\theta^*)\right)u + \tfrac{1}{6}\sum_i u_i u^T\left(\tfrac{\partial}{\partial\theta_i}\nabla^2\hat{\ell}_{CL}(\bar{\theta})\big|_{\bar{\theta}=\theta^*+\alpha u}\right)u, \tag{A.91}$$

where $\alpha \in [0,1]$. We now lower-bound the first-, second-, and third-order terms.

First-order term in Eq. (A.91): We can use Lemma A.1.4 to bound the first term with $(\nabla\hat{\ell}_{CL}(\theta^*))^T u \geq -\delta B$ with probability at least $1 - 2r\exp(-\frac{\delta^2 n}{2M_{max}^2\phi_{max}^2})$.

Second-order term in Eq. (A.91): Let u_{A_i} denote the elements of u corresponding to the component of the pseudolikelihood loss for Y_{A_i}; let r_{A_i} denote the length of u_{A_i}; and let M_{min} denote the minimum number of likelihood components in which any parameter participates.

$$\begin{aligned}
\tfrac{1}{2}u^T\left(\nabla^2\hat{\ell}_{CL}(\theta^*)\right)u &= \tfrac{1}{2}\sum_i u_{A_i}^T\left(\nabla^2[\hat{\ell}_{CL}(\theta^*)]_{A_i}\right)u_{A_i} && \text{(A.92)}\\
&\geq \tfrac{1}{2}\sum_i \Lambda_{min}\left(\nabla^2[\hat{\ell}_{CL}(\theta^*)]_{A_i}\right)\|u_{A_i}\|_2^2 && \text{(A.93)}
\end{aligned}$$

Continuing,

$$\begin{aligned}
\tfrac{1}{2}u^T\left(\nabla^2\hat{\ell}_{CL}(\theta^*)\right)u &\geq \tfrac{1}{2}\sum_i \Lambda_{min}\left(\nabla^2[\hat{\ell}_{CL}(\theta^*)]_{A_i}\right)\|u_{A_i}\|_2^2 && \text{(A.94)}\\
&= \tfrac{1}{2}\sum_t\left(\sum_{i\,:\,u_t\in u_{A_i}}\Lambda_{min}\left(\nabla^2[\hat{\ell}_{CL}(\theta^*)]_{A_i}\right)\right)u_t^2 && \text{(A.95)}\\
&= \tfrac{1}{2}\sum_t \hat{\rho}_t u_t^2 && \text{(A.96)}\\
&\geq \tfrac{1}{4}\sum_t \rho_t u_t^2 && \text{(A.97)}\\
&\geq \tfrac{1}{4}\rho_{min}\sum_t u_t^2 && \text{(A.98)}\\
&= \tfrac{1}{4}\rho_{min}\|u\|_2^2 && \text{(A.99)}\\
&\geq \tfrac{r^{-1}}{4}\rho_{min}\|u\|_1^2 && \text{(A.100)}\\
&= \tfrac{r^{-1}}{4}\rho_{min}B^2, && \text{(A.101)}
\end{aligned}$$

where we used Lemma A.1.5 to lower-bound $\hat{\rho}_t$ with $\rho_t/2$ with high probability.

Third-order term in Eq. (A.91):

$$\begin{aligned}
&\tfrac{1}{6}\sum_t u_t u^T\left(\tfrac{\partial}{\partial\theta_t}\nabla^2\hat{\ell}_{CL}(\bar{\theta})\big|_{\bar{\theta}=\theta^*+\alpha u}\right)u && \text{(A.102)}\\
&= \tfrac{1}{6}\sum_t u_t u^T\Bigg(\sum_{i\,:\,\theta_t\in\theta_{A_i}}\mathbf{E}\left[\phi_t\phi_{A_i}^{\otimes}\right] + 2\mathbf{E}\left[\phi_t\right]\mathbf{E}\left[\phi_{A_i}\right]^{\otimes} - \mathbf{E}\left[\phi_t\right]\mathbf{E}\left[\phi_{A_i}^{\otimes}\right] && \text{(A.103)}\\
&\qquad -\mathbf{E}\left[\phi_{A_i}\right]\mathbf{E}\left[\phi_t\phi_{A_i}\right]^T - \mathbf{E}\left[\phi_t\phi_{A_i}\right]\mathbf{E}\left[\phi_{A_i}\right]^T\Bigg)u, && \text{(A.104)}
\end{aligned}$$

where all expectations are w.r.t. $P_{\theta^*+\alpha u}(Y'_{A_i}|Y_{\backslash A_i},X)$, and $\phi_{A_i}=\phi_{A_i}(Y'_{A_i},Y_{\backslash A_i},X)$ and $\phi_t=\phi_t(Y'_{A_i},Y_{\backslash A_i},X)$. We can lower-bound and collapse the various terms on the right-hand side, just as in the proof of Theorem 2.3.1:

$$\tfrac{1}{6}\sum_t u_t u^T\left(\tfrac{\partial}{\partial\theta_t}\nabla^2\hat{\ell}_{CL}(\bar{\theta})\big|_{\bar{\theta}=\theta^*+\alpha u}\right)u \tag{A.105}$$

$$=\tfrac{1}{6}\sum_t u_t\sum_{i\,:\,\theta_t\in\theta_{A_i}}u_{A_i}^T\Bigg(\mathbf{E}\left[\phi_t\phi_{A_i}^{\otimes}\right]+2\mathbf{E}\left[\phi_t\right]\mathbf{E}\left[\phi_{A_i}\right]^{\otimes}-\mathbf{E}\left[\phi_t\right]\mathbf{E}\left[\phi_{A_i}^{\otimes}\right] \tag{A.106}$$

$$-\mathbf{E}\left[\phi_{A_i}\right]\mathbf{E}\left[\phi_t\phi_{A_i}\right]^T-\mathbf{E}\left[\phi_t\phi_{A_i}\right]\mathbf{E}\left[\phi_{A_i}\right]^T\Bigg)u_{A_i} \tag{A.107}$$

$$=\tfrac{1}{6}\sum_t u_t\sum_{i\,:\,\theta_t\in\theta_{A_i}}\Bigg(\mathbf{E}\left[\phi_t(u_{A_i}^T\phi_{A_i})^2\right]+2\mathbf{E}\left[\phi_t\right]\mathbf{E}\left[u_{A_i}^T\phi_{A_i}\right]^2-\mathbf{E}\left[\phi_t\right]\mathbf{E}\left[(u_{A_i}^T\phi_{A_i})^2\right] \tag{A.108}$$

$$-2\mathbf{E}\left[u_{A_i}^T\phi_{A_i}\right]\mathbf{E}\left[\phi_t u_{A_i}^T\phi_{A_i}\right]^T\Bigg)u_{A_i} \tag{A.109}$$

$$=\tfrac{1}{6}\sum_i\mathbf{E}\left[(u_{A_i}^T\phi_{A_i})^3\right]+2\mathbf{E}\left[u_{A_i}^T\phi_{A_i}\right]\mathbf{E}\left[u_{A_i}^T\phi_{A_i}\right]^2-3\mathbf{E}\left[u_{A_i}^T\phi_{A_i}\right]\mathbf{E}\left[(u_{A_i}^T\phi_{A_i})^2\right]. \tag{A.110}$$

Using Jensen's inequality multiple times, we can continue:

$$\tfrac{1}{6}\sum_i\mathbf{E}\left[(u_{A_i}^T\phi_{A_i})^3\right]+2\mathbf{E}\left[u_{A_i}^T\phi_{A_i}\right]\mathbf{E}\left[u_{A_i}^T\phi_{A_i}\right]^2-3\mathbf{E}\left[u_{A_i}^T\phi_{A_i}\right]\mathbf{E}\left[(u_{A_i}^T\phi_{A_i})^2\right] \tag{A.111}$$

$$\geq\tfrac{1}{6}\sum_i 3\mathbf{E}\left[u_{A_i}^T\phi_{A_i}\right]^3-3\mathbf{E}\left[u_{A_i}^T\phi_{A_i}\right]\mathbf{E}\left[(u_{A_i}^T\phi_{A_i})^2\right] \tag{A.112}$$

$$=\tfrac{1}{2}\sum_i\mathbf{E}\left[u_{A_i}^T\phi_{A_i}\right]^3-\mathbf{E}\left[u_{A_i}^T\phi_{A_i}\right]\mathbf{E}\left[(u_{A_i}^T\phi_{A_i})^2\right] \tag{A.113}$$

$$=-\tfrac{1}{2}\sum_i\left|\mathbf{E}\left[u_{A_i}^T\phi_{A_i}\right]\right|\cdot\left(\mathbf{E}\left[(u_{A_i}^T\phi_{A_i})^2\right]-\mathbf{E}\left[u_{A_i}^T\phi_{A_i}\right]^2\right) \tag{A.114}$$

$$\geq-\tfrac{1}{2}\sum_i\left|\mathbf{E}\left[u_{A_i}^T\phi_{A_i}\right]\right|\mathbf{E}\left[(u_{A_i}^T\phi_{A_i})^2\right] \tag{A.115}$$

$$\geq-\tfrac{1}{2}\sum_i\mathbf{E}\left[\left|u_{A_i}^T\phi_{A_i}\right|^3\right]. \tag{A.116}$$

Applying Holder's inequality, we can continue:

$$-\tfrac{1}{2}\sum_i\mathbf{E}\left[\left|u_{A_i}^T\phi_{A_i}\right|^3\right]\ \geq\ -\tfrac{1}{2}\sum_i\|u_{A_i}\|_1^3\phi_{max}^3\ \geq\ -\tfrac{1}{2}M_{max}\|u\|_1^3\phi_{max}^3\ =\ -\tfrac{1}{2}M_{max}B^3\phi_{max}^3. \tag{A.117}$$

Two of our bounds in this proof had small probabilities of failure. Using a union bound, we get the probability of at least one failing, finishing the proof. ■

Proof of Theorem 2.3.3: As in the proof of Theorem 2.3.1, we define $G:\mathbb{R}^r\rightarrow\mathbb{R}$ by

$$G(u)=\hat{\ell}_{CL}(\theta^*+u)-\hat{\ell}_{CL}(\theta^*)+\lambda\left(\|\theta^*+u\|_p-\|\theta^*\|_p\right), \tag{A.118}$$

with the difference that we now use the composite likelihood loss. As in Theorem 2.3.1, we wish to show that $G(u)>0$ for all $u\in\mathbb{R}^r$ with $\|u\|_1=B$ for some $B>0$, which will imply that $\|\hat{u}\|_1\leq B$.

Let $u\in\mathbb{R}^r$ with $\|u\|_1=B$. Using Lemma A.1.6, we can lower-bound G:

$$\begin{aligned}G(u)\ \ \geq\ \ &-\delta B+\tfrac{r^{-1}}{4}\rho_{min}B^2-\tfrac{1}{2}M_{max}\phi_{max}^3B^3\\&+\lambda\left(\|\theta^*+u\|_p-\|\theta^*\|_p\right),\end{aligned} \tag{A.119}$$

which holds with the probability given in Lemma A.1.6. As in the proof of Theorem 2.3.1, we can lower-bound the regularization term (for both L_1 and L_2 regularization): $\lambda\left(\|\theta^* + u\|_p - \|\theta^*\|_p\right) \geq -\lambda B$. Combining these bounds, we get:

$$\begin{aligned} G(u) &\geq -\delta B + \tfrac{r^{-1}}{4}\rho_{min} B^2 - \tfrac{1}{2} M_{max} B^3 \phi_{max}^3 - \lambda B && \text{(A.120)} \\ &= B\left[-\delta + \tfrac{r^{-1}}{4}\rho_{min} B - \tfrac{1}{2} M_{max} B^2 \phi_{max}^3 - \lambda\right]. && \text{(A.121)} \end{aligned}$$

Note that we need $\lambda > 0, B > 0, \delta > 0$. Eq. (A.121) will be strictly greater than 0 if $B > 0$ and

$$\lambda \; < \; -\delta + \tfrac{r^{-1}}{4}\rho_{min} B - \tfrac{1}{2} M_{max} B^2 \phi_{max}^3. \qquad \text{(A.122)}$$

Maximizing this bound w.r.t. B gives $B = \frac{\rho_{min}}{4r M_{max} \phi_{max}^3}$. However, we would like for B to shrink as $n^{-1/2}$, the asymptotic rate of convergence, so instead let

$$B = \frac{\rho_{min}}{4r M_{max} \phi_{max}^3} n^{-\xi/2}, \qquad \text{(A.123)}$$

where $\xi \in (0, 1)$. Plugging in this value for B gives

$$\lambda \; < \; -\delta + \frac{\rho_{min}^2}{2^4 r^2 M_{max} \phi_{max}^3} n^{-\xi/2} - \frac{\rho_{min}^2}{2^5 r^2 M_{max} \phi_{max}^3} n^{-\xi}. \qquad \text{(A.124)}$$

We want to choose δ to be large but still keep $\lambda > 0$, so choose

$$\lambda = \delta = \frac{\rho_{min}^2}{2^6 r^2 M_{max} \phi_{max}^3} n^{-\xi/2}, \qquad \text{(A.125)}$$

which makes Eq. (A.124) hold if $n > 1$. Now that we have chosen δ, we can simplify the probability of failure from Lemma A.1.6:

$$\begin{aligned} & 2r \exp\left(-\tfrac{\delta^2 n}{2 M_{max}^2 \phi_{max}^2}\right) + 2|\mathcal{A}| r^2 \exp\left(-\tfrac{n C_{min}^2}{2^5 r^2 \phi_{max}^4}\right) && \text{(A.126)} \\ &= 2r \exp\left(-\tfrac{\rho_{min}^4 n^{1-\xi}}{2^{13} r^4 M_{max}^4 \phi_{max}^8}\right) + 2|\mathcal{A}| r^2 \exp\left(-\tfrac{n C_{min}^2}{2^5 r^2 \phi_{max}^4}\right) && \text{(A.127)} \\ &\leq 2r \exp\left(-\tfrac{C_{min}^4 n^{1-\xi}}{2^{13} r^4 M_{max}^4 \phi_{max}^8}\right) + 2|\mathcal{A}| r^2 \exp\left(-\tfrac{n C_{min}^2}{2^5 r^2 \phi_{max}^4}\right) && \text{(A.128)} \\ &\leq 2r(|\mathcal{A}| r + 1) \exp\left(-\tfrac{C_{min}^4}{2^{13} r^4 M_{max}^4 \phi_{max}^8} n^{1-\xi}\right). && \text{(A.129)} \end{aligned}$$

The above bound is very loose in combining the two exponential terms; in particular, we would like ρ_{min} to remain in the bound. To derive a sufficient condition for this tighter combination, we first require that the left-hand term in the probability of failure be meaningful, i.e., at most 1:

$$\begin{aligned} 2r \exp\left(-\tfrac{\rho_{min}^4 n^{1-\xi}}{2^{13} r^4 M_{max}^4 \phi_{max}^8}\right) &\leq 1 && \text{(A.130)} \\ \log(2r) - \tfrac{\rho_{min}^4 n^{1-\xi}}{2^{13} r^4 M_{max}^4 \phi_{max}^8} &\leq 0 && \text{(A.131)} \end{aligned}$$

$$n^{1-\xi} \;\geq\; \frac{2^{13} r^4 M_{max}^4 \phi_{max}^8}{\rho_{min}^4} \log(2r). \qquad \text{(A.132)}$$

To keep ρ_{min} in the bound, we want to show that the left-hand exponential term in Eq. (A.127) dominates the right-hand term. Thus, we must use Eq. (A.132) to show a sufficient condition for:

$$2r\exp\left(-\frac{\rho_{min}^4 n^{1-\xi}}{2^{13}r^4M_{max}^4\phi_{max}^8}\right) \quad\geq\quad 2|\mathcal{A}|r^2\exp\left(-\frac{nC_{min}^2}{2^5r^2\phi_{max}^4}\right). \tag{A.133}$$

We may replace n with $n^{1-\xi}$ on the right-hand side. Since the left-hand term decreases more slowly in n than the right-hand term, we may replace n with the value from Eq. (A.132):

$$1 \quad\geq\quad 2|\mathcal{A}|r^2\exp\left(-\frac{2^8C_{min}^2r^2M_{max}^4\phi_{max}^4}{\rho^4}\log(2r)\right) \tag{A.134}$$

$$|\mathcal{A}| \quad\leq\quad \frac{1}{2r^2}\exp\left(\frac{2^8C_{min}^2r^2M_{max}^4\phi_{max}^4}{\rho^4}\log(2r)\right). \tag{A.135}$$

Since ρ_{min} is a sum of at most M_{max} eigenvalues, and since any eigenvalue is at most ϕ_{max}^2r (as shown in Theorem 2.3.1), we know $\rho \leq M_{max}\phi_{max}^2r$. Plugging this in, it suffices to show that:

$$|\mathcal{A}| \quad\leq\quad \frac{1}{2r^2}\exp\left(\frac{2^8C_{min}^2M_{max}^2}{\rho^2}\log(2r)\right) \tag{A.136}$$

$$= \quad \frac{1}{2r^2}(2r)^{\left\lfloor\frac{2^8C_{min}^2M_{max}^2}{\rho^2}\right\rfloor}. \tag{A.137}$$

Therefore, if we have

$$|\mathcal{A}| \leq \frac{1}{2r^2}(2r)^{\left\lfloor\frac{2^8C_{min}^2M_{max}^2}{\rho^2}\right\rfloor}, \tag{A.138}$$

then we know that the probability of failure is at most:

$$4r\exp\left(-\frac{\rho_{min}^4n^{1-\xi}}{2^{13}r^4M_{max}^4\phi_{max}^8}\right). \quad \blacksquare \tag{A.139}$$

Note that this bound is better than that from separate regressions, though both bounds are loose in terms of their treatments of shared parameters.

Proof of Corollary 2.3.4: If we wish to have a probability of failure of at most δ when we have n samples, we may choose ξ accordingly:

$$2r(|\mathcal{A}|r+1)\exp\left(-\frac{C_{min}^4}{2^{13}r^4M_{max}^4\phi_{max}^8}n^{1-\xi}\right) \quad\leq\quad \delta \tag{A.140}$$

$$\log(2r(|\mathcal{A}|r+1)) - \frac{C_{min}^4}{2^{13}r^4M_{max}^4\phi_{max}^8}n^{1-\xi} \quad\leq\quad \log\delta \tag{A.141}$$

$$\frac{C_{min}^4}{2^{13}r^4M_{max}^4\phi_{max}^8}n^{1-\xi} \quad\geq\quad \log\frac{2r(|\mathcal{A}|r+1)}{\delta} \tag{A.142}$$

$$n^{1-\xi} \quad\geq\quad \frac{2^{13}r^4M_{max}^4\phi_{max}^8}{C_{min}^4}\log\frac{2r(|\mathcal{A}|r+1)}{\delta} \tag{A.143}$$

$$(1-\xi)\log n \quad\geq\quad \log\frac{2^{13}r^4M_{max}^4\phi_{max}^8}{C_{min}^4} + \log\log\frac{2r(|\mathcal{A}|r+1)}{\delta} \tag{A.144}$$

$$(1-\xi) \quad\geq\quad \frac{1}{\log n}\left(\log\frac{2^{13}r^4M_{max}^4\phi_{max}^8}{C_{min}^4} + \log\log\frac{2r(|\mathcal{A}|r+1)}{\delta}\right) \tag{A.145}$$

$$\xi \quad\leq\quad 1 - \frac{1}{\log n}\left(\log\frac{2^{13}r^4M_{max}^4\phi_{max}^8}{C_{min}^4} + \log\log\frac{2r(|\mathcal{A}|r+1)}{\delta}\right) \tag{A.146}$$

$$\tag{A.147}$$

We will set ξ equal to this upper bound in the next part. Likewise, if we wish to have parameter estimation error at most ϵ, then we need:

$$\begin{aligned} \tfrac{\rho_{min}}{4rM_{max}\phi_{max}^3} n^{-\xi/2} &\leq \epsilon && \text{(A.148)} \\ \log \tfrac{\rho_{min}}{4rM_{max}\phi_{max}^3} - \tfrac{\xi}{2}\log n &\leq \log \epsilon. && \text{(A.149)} \end{aligned}$$

Rewrite the left-hand side:

$$\begin{aligned} & \log \tfrac{\rho_{min}}{4rM_{max}\phi_{max}^3} - \tfrac{\xi}{2}\log n && \text{(A.150)} \\ &= \log \tfrac{\rho_{min}}{4rM_{max}\phi_{max}^3} - \tfrac{1}{2}\left(\log n - \left(\log \tfrac{2^{13}r^4M_{max}^4\phi_{max}^8}{C_{min}^4} + \log\log \tfrac{2r(|\mathcal{A}|r+1)}{\delta}\right)\right) && \text{(A.151)} \\ &= \tfrac{1}{2}\left[\log \tfrac{\rho_{min}^2}{2^4r^2M_{max}^2\phi_{max}^6} - \log n + \log \tfrac{2^{13}r^4M_{max}^4\phi_{max}^8}{C_{min}^4} + \log\log \tfrac{2r(|\mathcal{A}|r+1)}{\delta}\right] && \text{(A.152)} \\ &= \tfrac{1}{2}\left[-\log n + \log \tfrac{2^9r^2M_{max}^2\phi_{max}^2\rho_{min}^2}{C_{min}^4} + \log\log \tfrac{2r(|\mathcal{A}|r+1)}{\delta}\right]. && \text{(A.153)} \end{aligned}$$

Recombine this left-hand side with Eq. (A.149):

$$-\log n + \log \frac{2^9r^2M_{max}^2\phi_{max}^2\rho_{min}^2}{C_{min}^4} + \log\log \frac{2r(|\mathcal{A}|r+1)}{\delta} \quad \leq \quad 2\log\epsilon \qquad \text{(A.154)}$$

$$\begin{aligned} \log n &\geq \log \frac{2^9r^2M_{max}^2\phi_{max}^2\rho_{min}^2}{C_{min}^4} + \log\log \frac{2r(|\mathcal{A}|r+1)}{\delta} - 2\log\epsilon && \text{(A.155)} \\ n &\geq \frac{2^9r^2M_{max}^2\phi_{max}^2\rho_{min}^2}{C_{min}^4}\frac{1}{\epsilon^2}\log \frac{2r(|\mathcal{A}|r+1)}{\delta} && \text{(A.156)} \end{aligned}$$

If, however, we assume that $|\mathcal{A}| \leq \frac{1}{2r^2}(2r)^{\left[\frac{2^8C_{min}^2M_{max}^2}{\rho^2}\right]}$, then our probability of failure changes, so we require:

$$\begin{aligned} 4r\exp\left(-\tfrac{\rho_{min}^4 n^{1-\xi}}{2^{13}r^4M_{max}^4\phi_{max}^8}\right) &\leq \delta && \text{(A.157)} \\ \log(4r) - \tfrac{\rho_{min}^4 n^{1-\xi}}{2^{13}r^4M_{max}^4\phi_{max}^8} &\leq \log\delta && \text{(A.158)} \\ \tfrac{\rho_{min}^4 n^{1-\xi}}{2^{13}r^4M_{max}^4\phi_{max}^8} &\geq \log\tfrac{4r}{\delta} && \text{(A.159)} \end{aligned}$$

$$\begin{aligned} n^{1-\xi} &\geq \tfrac{2^{13}r^4M_{max}^4\phi_{max}^8}{\rho_{min}^4}\log\tfrac{4r}{\delta} && \text{(A.160)} \\ \xi &\leq 1 - \tfrac{1}{\log n}\left(\log \tfrac{2^{13}r^4M_{max}^4\phi_{max}^8}{\rho_{min}^4} + \log\log\tfrac{4r}{\delta}\right). && \text{(A.161)} \end{aligned}$$

Plugging this value for ξ into Eq. (A.149), we get:

$$\begin{aligned} \log \tfrac{\rho_{min}}{4rM_{max}\phi_{max}^3} - \tfrac{1}{2}\left[1 - \tfrac{1}{\log n}\left(\log \tfrac{2^{13}r^4M_{max}^4\phi_{max}^8}{\rho_{min}^4} + \log\log\tfrac{4r}{\delta}\right)\right]\log n &\leq \log\epsilon && \text{(A.162)} \\ \log \tfrac{\rho_{min}^2}{2^4r^2M_{max}^2\phi_{max}^6} - \log n + \log \tfrac{2^{13}r^4M_{max}^4\phi_{max}^8}{\rho_{min}^4} + \log\log\tfrac{4r}{\delta} &\leq 2\log\epsilon && \text{(A.163)} \end{aligned}$$

$$\begin{aligned} \log n &\geq \log \tfrac{\rho_{min}^2}{2^4r^2M_{max}^2\phi_{max}^6} + \log \tfrac{2^{13}r^4M_{max}^4\phi_{max}^8}{\rho_{min}^4} + \log\log\tfrac{4r}{\delta} + \log\tfrac{1}{\epsilon^2} && \text{(A.164)} \\ &= \log \tfrac{2^9r^2M_{max}^2\phi_{max}^2}{\rho_{min}^2} + \log\log\tfrac{4r}{\delta} + \log\tfrac{1}{\epsilon^2} && \text{(A.165)} \\ n &\geq \tfrac{2^9r^2M_{max}^2\phi_{max}^2}{\rho_{min}^2}\tfrac{1}{\epsilon^2}\log\tfrac{4r}{\delta}. \quad \blacksquare && \text{(A.166)} \end{aligned}$$

A.1.4 Disjoint Optimization

Proof of Lemma 2.3.6: Let $M_t = |\{i \,:\, \theta_t \in \theta_{A_i}\}|$.

$$\begin{align}
\|\hat{\theta} - \theta^*\|_1 &= \sum_t \left|\hat{\theta}_t - \theta_t^*\right| \tag{A.167}\\
&= \sum_t \left| \tfrac{1}{M_t} \sum_{i\,:\,\theta_t \in \theta_{A_i}} \hat{\theta}_t^{(A_i)} - \theta^*{}_t \right| \tag{A.168}\\
&= \sum_t \tfrac{1}{M_t} \left| \sum_{i\,:\,\theta_t \in \theta_{A_i}} \hat{\theta}_t^{(A_i)} - \theta^*{}_t \right| \tag{A.169}\\
&\leq \sum_t \tfrac{1}{M_t} \sum_{i\,:\,\theta_t \in \theta_{A_i}} \left|\hat{\theta}_t^{(A_i)} - \theta^*{}_t\right| \tag{A.170}\\
&= \sum_i \sum_{t\,:\,\theta_t \in \theta_{A_i}} \tfrac{1}{M_t} \left|\hat{\theta}_t^{(A_i)} - \theta^*{}_t\right| \tag{A.171}\\
&\leq \sum_i \sum_{t\,:\,\theta_t \in \theta_{A_i}} \left|\hat{\theta}_t^{(A_i)} - \theta^*{}_t\right| \tag{A.172}\\
&\leq \sum_i \epsilon \tag{A.173}\\
&= |\mathcal{A}|\epsilon \quad \blacksquare \tag{A.174}
\end{align}$$

Proof of Theorem 2.3.7: Lemma 2.3.6 shows we can use Corollary 2.3.2 by shrinking the desired error ϵ and the probability of failure δ by factors of $1/|\mathcal{A}|$. A union bound combines the probabilities of failure. ■

A.1.5 Bounding the KL with Bounds on Parameter Estimation Error

This subsection uses bounds on the parameter estimation error to bound the log loss of our estimated distribution w.r.t. the target distribution.

The previous theorem demonstrates that there are two convergence regimes. Far from the optimum parameters, the log loss is approximately linear in the parameter estimation error. Close to the optimum, the log loss converges quadratically w.r.t. the parameter estimation error.

We prove two lemmas (for the two regimes) before proving the theorem.

Lemma A.1.7. ***(Third-Order Taylor Expansion)*** *Given a CRF factorizing as in Eq. (A.1) with parameters θ^* and maximum feature magnitude ϕ_{max}, assume that the maximum eigenvalue of the Hessian of the log loss at θ^* is Λ_{max}. Then the expected loss using a vector of parameters θ obeys the following bounds:*

$$\begin{align}
\ell_L(\theta) &\leq \ell_L(\theta^*) + \frac{\Lambda_{max}}{2}\|\theta - \theta^*\|_1^2 + \phi_{max}^3 \|\theta - \theta^*\|_1^3 \tag{A.175}\\
\ell_L(\theta) &\leq \ell_L(\theta^*) + \frac{\Lambda_{max}}{2}\|\theta - \theta^*\|_2^2 + \phi_{max}^3 r^{3/2} \|\theta - \theta^*\|_2^3. \tag{A.176}
\end{align}$$

The second-order term dominates when, respectively,

$$\|\theta - \theta^*\|_1 \leq \frac{\Lambda_{max}}{2\phi_{max}^3} \tag{A.177}$$

$$\|\theta - \theta^*\|_2 \leq \frac{\Lambda_{max}}{2r^{3/2}\phi_{max}^3}. \tag{A.178}$$

Proof: Write out the third-order Taylor expansion of the log loss in Eq. (A.2) w.r.t. θ around θ^*:

$$\begin{aligned} \ell_L(\theta) = \; & \ell_L(\theta^*) \\ & + \tfrac{1}{2}(\theta - \theta^*)^T \left(\nabla^2 \ell_L(\theta^*)\right)(\theta - \theta^*) \\ & + \tfrac{1}{6}\sum_i (\theta_i - \theta_i^*)(\theta - \theta^*)^T \left(\tfrac{\partial}{\partial \bar{\theta}_i} \nabla^2 \ell_L(\bar{\theta})|_{\bar{\theta}=\alpha\theta+(1-\alpha)\theta^*}\right)(\theta - \theta^*), \end{aligned} \tag{A.179}$$

where $\alpha \in [0,1]$. Note that the first-order term is 0. Let $u = \theta - \theta^*$. The second-order term may be upper-bounded using the maximum eigenvalue of the Hessian:

$$\tfrac{1}{2}u^T \left(\nabla^2 \ell_L(\theta^*)\right) u \leq \tfrac{1}{2}\Lambda_{max}\|u\|_2^2 \leq \tfrac{1}{2}\Lambda_{max}\|u\|_1^2. \tag{A.180}$$

The third-order term may be upper-bounded as well:

$$\tfrac{1}{6}\sum_i u_i u^T \left(\tfrac{\partial}{\partial \bar{\theta}_i} \nabla^2 \ell_L(\bar{\theta})|_{\bar{\theta}=\alpha\theta+(1-\alpha)\theta^*}\right) u \tag{A.181}$$

$$= \tfrac{1}{6}\sum_i u_i \Big(\mathbf{E}\left[\phi_i (u^T\phi)^2\right] + 2\mathbf{E}\left[\phi_i\right]\left(\mathbf{E}\left[u^T\phi\right]\right)^2 \tag{A.182}$$

$$-\mathbf{E}\left[\phi_i\right]\mathbf{E}\left[(u^T\phi)^2\right] - 2\mathbf{E}\left[u^T\phi\right]\mathbf{E}\left[\phi_i(u^T\phi)\right] \Big) \tag{A.183}$$

$$= \tfrac{1}{6}\left(\mathbf{E}\left[(u^T\phi)^3\right] + 2\mathbf{E}\left[u^T\phi\right]\left(\mathbf{E}\left[u^T\phi\right]\right)^2 - 3\mathbf{E}\left[u^T\phi\right]\mathbf{E}\left[(u^T\phi)^2\right]\right) \tag{A.184}$$

$$\leq \left(\mathbf{E}\left[|u^T\phi|\right]\right)^3 \tag{A.185}$$

$$\leq \left(\phi_{max}\|u\|_1\right)^3 \leq \left(\phi_{max}\sqrt{r}\|u\|_2\right)^3. \quad \blacksquare \tag{A.186}$$

Lemma A.1.8. ***(First-Order Taylor Expansion)*** *Given a CRF factorizing as in Eq. (A.1) with parameters θ^* and maximum feature magnitude ϕ_{max}, the expected loss using a vector of parameters θ obeys the following bounds:*

$$\ell_L(\theta) \leq \ell_L(\theta^*) + \phi_{max}\|\theta - \theta^*\|_1 \tag{A.187}$$

$$\ell_L(\theta) \leq \ell_L(\theta^*) + \phi_{max}\sqrt{r}\|\theta - \theta^*\|_2. \tag{A.188}$$

Proof: Write out the first-order Taylor expansion of the log loss in Eq. (A.2) w.r.t. θ around θ^*:

$$\ell_L(\theta) = \ell_L(\theta^*) + \left(\nabla \ell_L(\bar{\theta})|_{\bar{\theta}=\alpha\theta+(1-\alpha)\theta^*}\right)(\theta - \theta^*), \tag{A.189}$$

where $\alpha \in [0,1]$. We can upper-bound the first-order term using Holder's inequality:

$$\left(\nabla \ell_L(\bar{\theta})|_{\bar{\theta}=\alpha\theta+(1-\alpha)\theta^*}\right)(\theta - \theta^*),$$

$$= \left(\mathbf{E}_{P(X)}\left[-\mathbf{E}_{P_{\bar{\theta}}(Y|X)}\left[\phi(Y,X)\right] + \mathbf{E}_{P_\theta(Y'|X)}\left[\phi(Y',X)\right]\right]\right)(\theta - \theta^*) \tag{A.190}$$

$$\leq \left\|\mathbf{E}_{P(X)}\left[-\mathbf{E}_{P_{\bar{\theta}}(Y|X)}\left[\phi(Y,X)\right] + \mathbf{E}_{P_\theta(Y'|X)}\left[\phi(Y',X)\right]\right]\right\|_\infty \|\theta - \theta^*\|_1 \tag{A.191}$$

$$\leq \phi_{max}\|\theta - \theta^*\|_1 \leq \phi_{max}\sqrt{r}\|\theta - \theta^*\|_2. \quad \blacksquare \tag{A.192}$$

Proof of Theorem 2.3.5: Let $\delta = \|\theta - \theta^*\|_1$. Suppose the third-order bound is tighter; i.e.,

$$\frac{\Lambda_{max}}{2}\delta^2 + \phi_{max}^3\delta^3 \leq \phi\delta \tag{A.193}$$

$$\phi_{max}^3\delta^2 + \frac{\Lambda_{max}}{2}\delta - \phi \leq 0. \tag{A.194}$$

Solving, we get

$$\delta \leq \frac{-\frac{\Lambda_{max}}{2} + \sqrt{\frac{\Lambda_{max}^2}{4} + 4\phi_{max}^4}}{2\phi_{max}^3}. \tag{A.195}$$

Plugging this into the third-order bound, we can rewrite the third-order bound as:

$$\frac{\Lambda_{max}}{2}\delta^2 + \phi_{max}^3\delta^3 \leq \frac{\Lambda_{max}}{2}\delta^2 + \frac{-\frac{\Lambda_{max}}{2} + \sqrt{\frac{\Lambda_{max}^2}{4} + 4\phi_{max}^4}}{2}\delta^2 \tag{A.196}$$

$$= \frac{1}{2}\left(\frac{\Lambda_{max}}{2} + \sqrt{\frac{\Lambda_{max}^2}{4} + 4\phi_{max}^4}\right)\delta^2 \tag{A.197}$$

$$\leq \frac{1}{2}\left(\frac{\Lambda_{max}}{2} + \frac{\Lambda_{max}}{2} + 2\phi_{max}^2\right)\delta^2 \tag{A.198}$$

$$\leq \left(\frac{\Lambda_{max}}{2} + \phi_{max}^2\right)\delta^2. \quad \blacksquare \tag{A.199}$$

We discuss sample complexity bounds. The key is to establish a bound on the number of samples required to be in the quadratic convergence regime, after which the proof is trivial. To guarantee that Eq. (2.15) holds, we can use Corollary 2.3.4 to show it suffices to have:

$$\frac{2^9 r^2 M_{max}^2 \phi_{max}^2}{\rho_{min}^2}\frac{1}{n}\log\frac{4r}{\delta} \leq \left[\frac{-\frac{\Lambda_{max}}{2} + \sqrt{\frac{\Lambda_{max}^2}{4} + r\phi_{max}^4}}{2\phi_{max}^3}\right]^2 \tag{A.200}$$

$$= \frac{\frac{\Lambda_{max}^2}{4} + \frac{\Lambda_{max}^2}{4} + r\phi_{max}^4 - \Lambda_{max}\sqrt{\frac{\Lambda_{max}^2}{4} + r\phi_{max}^4}}{4\phi_{max}^6} \tag{A.201}$$

$$= \frac{\Lambda_{max}^2 + 2r\phi_{max}^4 - \Lambda_{max}\sqrt{\Lambda_{max}^2 + 4r\phi_{max}^4}}{8\phi_{max}^6} \tag{A.202}$$

$$n \geq \frac{2^{12} r^2 M_{max}^2 \phi_{max}^8}{\rho_{min}^2\left(\Lambda_{max}^2 + 2r\phi_{max}^4 - \Lambda_{max}\sqrt{\Lambda_{max}^2 + 4r\phi_{max}^4}\right)}\log\frac{4r}{\delta} \tag{A.203}$$

In this quadratic regime, to achieve log loss $\ell_L(\theta^*) + \epsilon$, we need:

$$\left(\frac{\Lambda_{max}}{2} + \phi_{max}^2\right)\frac{2^9 r^2 M_{max}^2 \phi_{max}^2}{\rho_{min}^2}\frac{1}{n}\log\frac{4r}{\delta} \leq \epsilon \tag{A.204}$$

$$n \geq \frac{2^8 r^2 M_{max}^2 \phi_{max}^2(\Lambda_{max} + 2\phi_{max}^2)}{\rho_{min}^2}\frac{1}{\epsilon}\log\frac{4r}{\delta}. \tag{A.205}$$

Likewise, in the linear regime, we need:

$$n \geq \frac{2^9 r^2 M_{max}^2 \phi_{max}^3}{\rho_{min}^2}\frac{1}{\epsilon}\log\frac{4r}{\delta}. \tag{A.206}$$

A.2 Canonical Parametrization: Proofs from Sec. 2.6

A.2.1 Proof of Theorem 2.6.14

We hide $\mathcal{X}$ in this proof since it does not play a role.

In Eq. (2.50), we can write each local conditional probability in terms of its factorization according to Eq. (2.49), where we write Z_i to denote the partition function for the local conditional distribution:

$$\begin{aligned}
&\log f_j^*(Y_{C_j^*}) && \text{(A.207)}\\
&= \sum_{U \subseteq C_j^*} (-1)^{|C_j^* \setminus U|} \log P\left(Y_{i_j}[Y_U, \bar{y}_{\setminus U}] \,\middle|\, MB_{Y_{i_j}}[Y_U, \bar{y}_{\setminus U}]\right) && \text{(A.208)}\\
&= \sum_{U \subseteq C_j^*} (-1)^{|C_j^* \setminus U|} \left[-Z_{i_j}\left(MB_{Y_{i_j}}[Y_U, \bar{y}_{\setminus U}]\right) + \sum_{k\,:\, i_j \in C_k} \log \phi_k^{(i_j)}(Y_U, \bar{y}_{\setminus U}) \right]. && \text{(A.209)}
\end{aligned}$$

There are two terms in the brackets: the partition function and a sum over factors. Consider the partition function only, and examine the sum over U. The partition function does not depend on Y_{i_j}, so a set U containing Y_{i_j} and a set U' not containing Y_{i_j} will produce the same value for Z_i. Since $Y_{i_j} \in C_j^*$, we may pair up subsets U of C_j^* s.t. each pair contains one set U and one set $U' = U \cup \{Y_{i_j}\}$. Substituting Eq. (A.209) into Eq. (2.51) and removing the local partition functions gives us:

$$\begin{aligned}
P(Y) \quad &\propto \quad \prod_{j=1}^{J^*} \exp\left(\sum_{U \subseteq C_j^*} (-1)^{|C_j^* \setminus U|} \sum_{k\,:\, i_j \in C_k} \log \phi_k^{(i_j)}(Y_U, \bar{y}_{\setminus U}) \right) && \text{(A.210)}\\
&= \quad \exp\left(\sum_{j=1}^{J^*} \sum_{U \subseteq C_j^*} (-1)^{|C_j^* \setminus U|} \sum_{k\,:\, i_j \in C_k} \log \phi_k^{(i_j)}(Y_U, \bar{y}_{\setminus U}) \right). && \text{(A.211)}
\end{aligned}$$

Suppose that we have no conflicting factor estimates; then for each k, $\phi_k^{(i_j)}$ are equal for all i_j, so the estimate in Eq. (A.211) equals the pseudolikelihood estimate (as proven in Theorem 2.6.13).

Now, recall that we assumed that there exists a factor $\phi_k(Y_{C_k}, X_{D_k})$ with at least two arguments in $\mathcal{Y}$; let $i \in C_k$ index one of these arguments. One canonical factor j is defined by $C_j^* = \{i\}$; For this canonical factor, we must let $i_j = i$. All other canonical factors j' must include arguments other than i in $C_{j'}^*$; for these canonical factors, let $i_j \neq i$.

The terms in the exponential in Eq. (A.211) corresponding to this original factor k and this canonical factor j (and its $i_j = i$) are:

$$\sum_{U \subseteq C_j^*} (-1)^{|C_j^* \setminus U|} \log \phi_k^{(i_j)}(Y_U, \bar{y}_{\setminus U}) \quad = \quad \log \phi_k^{(i)}(Y_i, \bar{y}_{\setminus i}) - \log \phi_k^{(i)}(\bar{y}), \qquad \text{(A.212)}$$

where we used $C_j^* = \{i\}$. Note that our choices of i_j for the other canonical factors j imply that these terms are the only ones in Eq. (A.211) which involve $\phi_k^{(i_j)}$. Keeping all other factor estimates non-conflicting, modify the value of $\phi_k^{(i)}(Y_i, \bar{y}_{\setminus i})$ to an arbitrary conflicting value for each value of Y_i. Since $\phi_k^{(i)}(Y_i, \bar{y}_{\setminus i})$ does not appear anywhere else in the canonical parametrization, the canonical parametrization estimate of $P(\mathcal{Y}|\mathcal{X})$ must have changed for at least one value of Y_i. Therefore, the canonical parametrization and the pseudolikelihood estimates are no longer equal. ■

A.2.2 Proof of Theorem 2.6.16

We hide $\mathcal{X}$ in this proof since it does not play a role.

Beginning with Eq. (2.56), rewrite each local conditional probability in terms of its factorization according to Eq. (2.49), where we write Z_i to denote the partition function for the local conditional:

$$\begin{aligned}
&\frac{1}{|C_j^*|}\sum_{i\in C_j^*}\sum_{U\subseteq C_j^*}(-1)^{|C_j^*\setminus U|}\log P\left(Y_i[Y_U,\overline{y}_{\setminus U}]\,\middle|\,Y_{N_i}[Y_U,\overline{y}_{\setminus U}]\right) && \text{(A.213)}\\
&=\frac{1}{|C_j^*|}\sum_{i\in C_j^*}\sum_{U\subseteq C_j^*}(-1)^{|C_j^*\setminus U|}\left[-Z_i\left(Y_{N_i}[Y_U,\overline{y}_{\setminus U}]\right)+\sum_{k\,:\,i\in C_k}\log\phi_k^{(i)}(Y_U,\overline{y}_{\setminus U})\right]. && \text{(A.214)}
\end{aligned}$$

There are two terms in the brackets: the partition function and a sum over factors. Consider the partition function only, and examine the sum over U. The partition function does not depend on Y_i, so a set U containing Y_i and a set U' not containing Y_i will produce the same value for Z_i. Since $Y_i \in C_j^*$, we may pair up subsets U of C_j^* s.t. each pair contains one set U and one set $U' = U \cup \{Y_i\}$. Substituting Eq. (A.214) into Eq. (2.51) and removing the local partition functions gives us:

$$\begin{aligned}
P(Y) &= P(\overline{y})\prod_{j=1}^{J^*}\exp\left(\frac{1}{|C_j^*|}\sum_{i\in C_j^*}\sum_{U\subseteq C_j^*}(-1)^{|C_j^*\setminus U|}\sum_{k\,:\,i\in C_k}\log\phi_k^{(i)}(Y_U,\overline{y}_{\setminus U})\right) && \text{(A.215)}\\
&= P(\overline{y})\exp\left(\sum_{j=1}^{J^*}\frac{1}{|C_j^*|}\sum_{i\in C_j^*}\sum_{U\subseteq C_j^*}(-1)^{|C_j^*\setminus U|}\sum_{k\,:\,i\in C_k}\log\phi_k^{(i)}(Y_U,\overline{y}_{\setminus U})\right) && \text{(A.216)}\\
&= P(\overline{y})\exp\left(\sum_{i}\sum_{j\,:\,i\in C_j^*}\frac{1}{|C_j^*|}\sum_{U\subseteq C_j^*}(-1)^{|C_j^*\setminus U|}\sum_{k\,:\,i\in C_k}\log\phi_k^{(i)}(Y_U,\overline{y}_{\setminus U})\right) && \text{(A.217)}\\
&= P(\overline{y})\exp\left(\sum_{i}\sum_{k\,:\,i\in C_k}\sum_{j\,:\,i\in C_j^*}\frac{1}{|C_j^*|}\sum_{U\subseteq C_j^*}(-1)^{|C_j^*\setminus U|}\log\phi_k^{(i)}(Y_U,\overline{y}_{\setminus U})\right) && \text{(A.218)}\\
&= P(\overline{y})\exp\left(\sum_{k}\sum_{i\in C_k}\sum_{j\,:\,i\in C_j^*}\frac{1}{|C_j^*|}\sum_{U\subseteq C_j^*}(-1)^{|C_j^*\setminus U|}\log\phi_k^{(i)}(Y_U,\overline{y}_{\setminus U})\right). && \text{(A.219)}
\end{aligned}$$

Consider the sum

$$\sum_{j\,:\,i\in C_j^*}\frac{1}{|C_j^*|}\sum_{U\subseteq C_j^*}(-1)^{|C_j^*\setminus U|}\log\phi_k^{(i)}(Y_U,\overline{y}_{\setminus U}). \tag{A.220}$$

If a given j corresponds to a C_j^* which contains an index t not appearing in C_k (i.e., $t \in C_j^*$ and $t \notin C_k$), then

$$\sum_{U\subseteq C_j^*}(-1)^{|C_j^*\setminus U|}\log\phi_k^{(i)}(Y_U,\overline{y}_{\setminus U}) = \sum_{U\subseteq(C_j^*\setminus\{t\})}(-1)^{|C_j^*\setminus U|}\left(\log\phi_k^{(i)}(Y_U,\overline{y}_{\setminus U})-\log\phi_k^{(i)}(Y_{U\cup\{t\}},\overline{y}_{\setminus(U\cup\{t\})})\right). \tag{A.221}$$

Since $t \notin C_k$, it is not an argument of the function $\phi_k^{(i)}$, so the two log terms cancel out. Therefore, we may assume that the sum over C_j^* is only over subsets of C_k which contain i:

$$P(Y) \quad\propto\quad \exp\left(\sum_k\sum_{i\in C_k}\sum_{C^*\subseteq(C_k\setminus\{i\})}\tfrac{1}{|C^*|+1}\sum_{U\subseteq(C^*\cup\{i\})}(-1)^{|(C^*\cup\{i\})\setminus U|}\log\phi_k^{(i)}(Y_U,\overline{y}_{\setminus U})\right). \tag{A.222}$$

We can swap the sums over C^* and U:

$$\sum_{C^*\subseteq(C_k\setminus\{i\})} \tfrac{1}{|C^*|+1} \sum_{U\subseteq(C^*\cup\{i\})} (-1)^{|(C^*\cup\{i\})\setminus U|} \log \phi_k^{(i)}(Y_U, \overline{y}_{\setminus U}) \tag{A.223}$$

$$= \sum_{C^*\subseteq(C_k\setminus\{i\})} \tfrac{1}{|C^*|+1} \sum_{U\subseteq C^*} (-1)^{|C^*\setminus U|} \left[\log \phi_k^{(i)}(Y_{U\cup i}, \overline{y}_{\setminus U\setminus i}) - \log \phi_k^{(i)}(Y_U, \overline{y}_i, \overline{y}_{\setminus U\setminus i})\right] \tag{A.224}$$

$$= \sum_{U\subseteq(C_k\setminus i)} \left(\sum_{C^{**}\subseteq(C_k\setminus i)\setminus U} \tfrac{1}{|C^{**}|+|U|+1}(-1)^{|C^{**}|} \right) \left[\log \phi_k^{(i)}(Y_{U\cup i}, \overline{y}_{\setminus U\setminus i}) - \log \phi_k^{(i)}(Y_U, \overline{y}_i, \overline{y}_{\setminus U})\right] \tag{A.225}$$

where we are setting $C^* = U \cup C^{**}$. Abbreviate $w(i,k,U) = \sum_{C^{**}\subseteq(C_k\setminus i)\setminus U} \frac{1}{|C^{**}|+|U|+1}(-1)^{|C^{**}|}$, and plug Eq. (A.225) back into Eq. (A.222).

$$P(Y) \quad \propto \quad \exp\left(\sum_k \sum_{i\in C_k} \sum_{U\subseteq(C_k\setminus i)} w(i,k,U) \left[\log \phi_k^{(i)}(Y_{U\cup i}, \overline{y}_{\setminus U\setminus i}) - \log \phi_k^{(i)}(Y_U, \overline{y}_i, \overline{y}_{\setminus U})\right] \right) \tag{A.226}$$

Recall that the pseudolikelihood estimate with disjoint regressions and factor averaging is:

$$P(Y) \quad \propto \quad \exp\left(\sum_k \sum_{i\in C_k} \tfrac{1}{|C_k|} \log \phi_k^{(i)}(Y_{C_k}) \right) . \tag{A.227}$$

Since we assumed that factor estimates may conflict in Eq. (A.226), we can adjust values of $\phi_k^{(i)}$ arbitrarily for any k, i, U and values of Y, as in the proof of Theorem 2.6.14. Therefore, in order for the right-hand side of Eq. (A.226) to be proportional to the right-hand side of Eq. (A.227), we would need

$$\sum_{U\subseteq(C_k\setminus i)} w(i,k,U) \left[\log \phi_k^{(i)}(Y_{U\cup i}, \overline{y}_{\setminus U\setminus i}) - \log \phi_k^{(i)}(Y_U, \overline{y}_i, \overline{y}_{\setminus U})\right] \quad = \quad \tfrac{1}{|C_k|} \log \phi_k^{(i)}(Y_{C_k}) \tag{A.228}$$

to hold for all k, $i \in C_k$. Fix k, $i \in C_k$. Pick a $U \subseteq C_k \setminus i$ s.t. $w(i,k,U) \neq 0$. We can adjust the values $\phi_k^{(i)}(Y_{U\cup i}, \overline{y}_{\setminus U\setminus i})$, $\phi_k^{(i)}(Y_U, \overline{y}_i, \overline{y}_{\setminus U})$ arbitrarily, so we can shift them s.t. the canonical parametrization and pseudolikelihood estimates of $\widehat{P}(\mathcal{Y}|\mathcal{X})$ are different, just as in the proof of Theorem 2.6.14. ■

Additional Calculations

Consider the sum over C^{**} in $w(i,k,U)$:

$$\sum_{C^{**}\subseteq(C_k\setminus i)\setminus U} \tfrac{1}{|C^{**}|+|U|+1}(-1)^{|C^{**}|} \quad = \quad \sum_{n=0}^{|(C_k\setminus i)\setminus U|} \tbinom{|(C_k\setminus i)\setminus U|}{n} \tfrac{1}{n+|U|+1}(-1)^n . \tag{A.229}$$

Using the identity $\sum_{k=0}^{n} \binom{n}{k} \frac{1}{k+r}(-1)^k = \binom{n+r}{r}^{-1}/r$ from Gould [2010], where we let $n = |(C_k \setminus i) \setminus U|$ and $r = |U| + 1$, this sum becomes:

$$\sum_{n=0}^{|(C_k\setminus i)\setminus U|} \tbinom{|(C_k\setminus i)\setminus U|}{n} \tfrac{1}{n+|U|+1}(-1)^n \quad = \quad \tbinom{|C_k|}{|U|+1}^{-1} \tfrac{1}{|U|+1} . \tag{A.230}$$

Plugging this into Eq. (A.222) and Eq. (A.225) gives:

$$P(Y) \quad \propto \quad \exp\left(\sum_k \sum_{i\in C_k} \sum_{U\subseteq(C_k\setminus i)} \tbinom{|C_k|}{|U|+1}^{-1} \tfrac{1}{|U|+1} \left[\log \phi_k^{(i)}(Y_{U\cup i}, \overline{y}_{\setminus U\setminus i}) - \log \phi_k^{(i)}(Y_U, \overline{y}_i, \overline{y}_{\setminus U})\right] \right) \tag{A.231}$$

Note $\binom{|C_k|}{|U|+1}^{-1} \frac{1}{|U|+1} = \frac{1}{|C_k|}$ when $U = \emptyset$.

A.2.3 Proof of Theorem 2.6.17

Instantiate Theorem 2.6.15 (averaged i_j) for pairwise models with $\mathcal{X} = \emptyset$:

$$\begin{aligned}\log P(Y) \quad = \quad & \log P(\overline{y}) + \sum_i \left(1 - \tfrac{|N_i|}{2}\right) \left[\log P(Y_i|\overline{y}_{N_i}) - \log P(\overline{y}_i|\overline{y}_{N_i})\right] \\ & + \tfrac{1}{2} \sum_{j \in N_i} \log P(Y_i|Y_j, \overline{y}_{N_i \setminus j}) - \log P(\overline{y}_i|Y_j, \overline{y}_{N_i \setminus j}) \,.\end{aligned} \tag{A.232}$$

Take the difference between the canonical parametrizations for $P(Y)$ (as in Eq. (A.232)) and $Q(Y)$ (as in Eq. (A.232), but with P, Q switched.

$$\begin{aligned}& \log Q(Y) - \log P(Y) \\ & \quad = \log Q(\overline{y}) - \log P(\overline{y}) \\ & \quad\quad + \sum_i \left(1 - \tfrac{|N_i|}{2}\right) \left[\log Q(Y_i|\overline{y}_{N_i}) - \log P(Y_i|\overline{y}_{N_i}) - \log Q(\overline{y}_i|\overline{y}_{N_i}) + \log P(\overline{y}_i|\overline{y}_{N_i})\right] \\ & \quad\quad\quad + \tfrac{1}{2} \sum_{j \in N_i} \log Q(Y_i|Y_j, \overline{y}_{N_i \setminus j}) - \log P(Y_i|Y_j, \overline{y}_{N_i \setminus j}) - \log Q(\overline{y}_i|Y_j, \overline{y}_{N_i \setminus j}) + \log P(\overline{y}_i|Y_j, \overline{y}_{N_i \setminus j}) \,.\end{aligned} \tag{A.233}$$

Apply Theorem 2.6.12 (reference assignment averaging) by taking the expectation over $\overline{y}$ w.r.t. $P(\overline{Y})$. Also, take the expectation over Y w.r.t. $Q(Y)$. Letting $P_i \doteq P(Y_i|Y_{N_i})$ and $Q_i \doteq Q(Y_i|Y_{N_i})$, we get:

$$\begin{aligned}& KL(Q\|P) + KL(P\|Q) \\ & \quad = \sum_i \left(1 - \tfrac{|N_i|}{2}\right) \mathbf{E}_{P(Y_{N_i})} \Big[\mathbf{E}_{Q(Y_i)} \left[\log Q_i - \log P_i\right] - \mathbf{E}_{P(Y_i|Y_{N_i})} \left[\log Q_i - \log P_i\right]\Big] \\ & \quad\quad + \tfrac{1}{2} \sum_{j \in N_i} \mathbf{E}_{Q(Y_j)P(Y_{N_i \setminus j})} \Big[\mathbf{E}_{Q(Y_i|Y_j)} \left[\log Q_i - \log P_i\right] - \mathbf{E}_{P(Y_i|Y_{N_i \setminus j})} \left[\log Q_i - \log P_i\right]\Big] \,.\end{aligned} \tag{A.234}$$

Rearranging terms gives the theorem's result:

$$\begin{aligned}& KL(P\|Q) + KL(Q\|P) \\ & \quad = \sum_i \left(1 - \tfrac{|N_i|}{2}\right) \mathbf{E}_{P(Y_{N_i})} \Big[\mathbf{E}_{P(Y_i|Y_{N_i})} \left[\log P_i - \log Q_i\right] + \mathbf{E}_{Q(Y_i)} \left[\log Q_i - \log P_i\right]\Big] \\ & \quad\quad + \tfrac{1}{2} \sum_{j \in N_i} \mathbf{E}_{Q(Y_j)P(Y_{N_i \setminus j})} \Big[\mathbf{E}_{P(Y_i|Y_{N_i \setminus j})} \left[\log P_i - \log Q_i\right] + \mathbf{E}_{Q(Y_i|Y_j)} \left[\log Q_i - \log P_i\right]\Big] \,.\end{aligned} \tag{A.235}$$

■

Appendix B

Parallel Regression

B.1 Proofs

B.1.1 Detailed Proofs: β for Squared Error and Logistic Loss

Assumption 4.2.1 and Assumption 4.3.1 both upper bound the change in objective from updating $\mathbf{x}$ with $\Delta\mathbf{x}$. We show how to do so for Assumption 4.3.1, which generalizes Assumption 4.2.1. For both losses, we upper-bound the objective using a second-order Taylor expansion of F around $\mathbf{x}$. Note that we do not use duplicated features in this section, but that does not affect the resulting β values.

Proof: β for Squared Error

The first and second derivatives of the L_1-regularized squared error (with duplicated features so weights are non-negative) are:

$$\begin{aligned} \nabla F(\mathbf{x}) &= \mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{A}^T\mathbf{y} + \lambda\mathbf{1} && \text{(B.1)} \\ \nabla^2 F(\mathbf{x}) &= \mathbf{A}^T\mathbf{A}\,, && \text{(B.2)} \end{aligned}$$

where $\mathbf{1}$ is an all-ones vector of the appropriate size. Note that, since derivatives of Eq. (4.2) of order higher than two are zero, the second order Taylor expansion is exact:

$$F(\mathbf{x}+\Delta\mathbf{x}) = F(\mathbf{x}) + (\Delta\mathbf{x})^T\nabla F(\mathbf{x}) + \tfrac{1}{2}(\Delta\mathbf{x})^T\nabla^2 F(\mathbf{x})\Delta\mathbf{x} \tag{B.3}$$

Plugging in the second order derivative gives $\beta = 1$:

$$F(\mathbf{x}+\Delta\mathbf{x}) = F(\mathbf{x}) + (\Delta\mathbf{x})^T\nabla F(\mathbf{x}) + \tfrac{1}{2}(\Delta\mathbf{x})^T\mathbf{A}^T\mathbf{A}\Delta\mathbf{x}\,. \tag{B.4}$$

This bound is exact for squared error but not for all losses.

Proof: β for Logistic Loss

Define $p_i = \frac{1}{1+\exp\left(-y_i \mathbf{a}_i^T \mathbf{x}\right)}$, the class conditional probability of y_i given $\mathbf{a}_i$. The first and second derivative of the logistic loss with L_1 regularization are:

$$\begin{aligned} \tfrac{\partial}{\partial x_j} F(\mathbf{x}) &= \lambda + \sum_{i=1}^{n} y_i \mathbf{A}_{ij} \left(p_i - 1\right) && \text{(B.5)} \\ \tfrac{\partial^2}{\partial x_j \partial x_k} F(\mathbf{x}) &= \sum_{i=1}^{n} \mathbf{A}_{ij} \mathbf{A}_{ik} \left(1 - p_i\right) p_i \,. && \text{(B.6)} \end{aligned}$$

Taylor's theorem tells us that there exists an $\hat{\mathbf{x}}$ s.t.

$$F(\mathbf{x} + \Delta\mathbf{x}) \leq F(\mathbf{x}) + (\Delta\mathbf{x})^T \nabla F(\mathbf{x}) + \tfrac{1}{2} (\Delta\mathbf{x})^T (\nabla^2 F(\hat{\mathbf{x}})) \Delta\mathbf{x} \,. \tag{B.7}$$

The second-order term is maximized by setting $p_i = \frac{1}{2}$ in the second derivative $\frac{\partial^2}{\partial x_j \partial x_k} F(\mathbf{x})$ for each j, k. Plugging in this value gives our bound with $\beta = \frac{1}{4}$:

$$F(\mathbf{x} + \Delta\mathbf{x}) \leq F(\mathbf{x}) + (\Delta\mathbf{x})^T \nabla F(\mathbf{x}) + \frac{1}{2} \frac{(\Delta\mathbf{x})^T \mathbf{A}^T \mathbf{A} \Delta\mathbf{x}}{4} \,. \tag{B.8}$$

B.1.2 Duplicated Features

Our work, like Shalev-Schwartz and Tewari [2009], uses duplicated features (with $\mathbf{x} \in \mathbb{R}^{2d}$ and $\mathbf{A} \in \mathbb{R}^{n \times 2d}$), but our actual implementation does not ($\mathbf{x} \in \mathbb{R}_+^d$ and $\mathbf{A} \in \mathbb{R}^{n \times d}$). They point out that the optimization problems with and without duplicated features are equivalent.

To see this, consider the form of $F(\mathbf{x})$ in Eq. (4.4). x_j only appears in the dot product $\hat{\mathbf{a}}_i^T \mathbf{x}$ via $\mathbf{A}_{i,j} x_j$, and x_{d+j} only appears via $-\mathbf{A}_{i,j} x_{d+j}$, where $\mathbf{A}$ is the original design matrix without duplicated features. Suppose $x_j > 0$ and $x_{d+j} > 0$, and assume w.l.o.g. that $x_j > x_{d+j}$. Then setting $x_j \longleftarrow x_j - x_{d+j}$ and $x_{d+j} \longleftarrow 0$ would give the same value for the loss term $L(\hat{\mathbf{a}}_i^T \mathbf{x},\, y_i)$, and it would decrease the regularization penalty by $2\lambda x_{d+j}$. Therefore, at the optimum, at most one of x_j, x_{d+j} will be non-zero, and the objectives with and without duplicated features will be equal.

B.1.3 Detailed Proof: Theorem 4.2.1

Define a potential function, where $\mathbf{x}^*$ is the optimal weight vector:

$$\Psi(\mathbf{x}) = \frac{\beta}{2} \|\mathbf{x} - \mathbf{x}^*\|_2^2 + F(\mathbf{x}) \,. \tag{B.9}$$

Claim: After updating weight x_j with δx_j,

$$\Psi(\mathbf{x}) - \Psi(\mathbf{x} + \Delta\mathbf{x}) \geq (x_j - x_j^*)(\nabla F(\mathbf{x}))_j \,. \tag{B.10}$$

To see this:

$$\begin{aligned}
\Psi(\mathbf{x}) - \Psi(\mathbf{x}+\Delta\mathbf{x}) &= \tfrac{\beta}{2}\left[\|\mathbf{x}-\mathbf{x}^*\|_2^2 - \|\mathbf{x}+\Delta\mathbf{x}-\mathbf{x}^*\|_2^2\right] + F(\mathbf{x}) - F(\mathbf{x}+\Delta\mathbf{x}) && \text{(B.11)}\\
&= -\tfrac{\beta}{2}\left[2(\mathbf{x}-\mathbf{x}^*)^T\Delta\mathbf{x} + (\delta x_j)^2\right] + F(\mathbf{x}) - F(\mathbf{x}+\Delta\mathbf{x}) && \text{(B.12)}\\
&\geq \beta\left(-\mathbf{x}^T\Delta\mathbf{x} + \mathbf{x}^{*T}\Delta\mathbf{x} - \tfrac{(\delta x_j)^2}{2}\right) - (\Delta\mathbf{x})^T\nabla F(\mathbf{x}) - \tfrac{\beta}{2}(\delta x_j)^2 && \text{(B.13)}\\
&= \beta\left(-x_j\delta x_j + x_j^*\delta x_j - (\delta x_j)^2\right) - \delta x_j(\nabla F(\mathbf{x}))_j && \text{(B.14)}\\
&\geq \beta\left(-x_j\delta x_j - (\delta x_j)^2\right) - x_j^*(\nabla F(\mathbf{x}))_j - \delta x_j(\nabla F(\mathbf{x}))_j\,. && \text{(B.15)}
\end{aligned}$$

Above, Eq. (B.13) used Assumption 4.2.1. Eq. (B.15) used the update rule for choosing δx_j in Eq. (4.5). Now there are two possible cases stemming from the update rule. Case 1: If $\delta x_j = -x_j$, then Eq. (B.15) simplifies to

$$\Psi(\mathbf{x}) - \Psi(\mathbf{x}+\Delta\mathbf{x}) \geq (x_j - x_j^*)(\nabla F(\mathbf{x}))_j\,. \qquad \text{(B.16)}$$

Case 2: If $\delta x_j = -(\nabla F(\mathbf{x}))_j/\beta$, then Eq. (B.15) again simplifies to

$$\begin{aligned}
\Psi(\mathbf{x}) - \Psi(\mathbf{x}+\Delta\mathbf{x}) &\geq x_j(\nabla F(\mathbf{x}))_j - \beta(\delta x_j)^2 - x_j^*(\nabla F(\mathbf{x}))_j + \beta(\delta x_j)^2 && \text{(B.17)}\\
&= (x_j - x_j^*)(\nabla F(\mathbf{x}))_j\,. && \text{(B.18)}
\end{aligned}$$

Having proven our claim, we can now take the expectation of Eq. (B.10) w.r.t. j, the chosen weight:

$$\begin{aligned}
\mathbf{E}\left[\Psi(\mathbf{x}) - \Psi(\mathbf{x}+\Delta\mathbf{x})\right] &\geq \mathbf{E}\left[(x_j - x_j^*)(\nabla F(\mathbf{x}))_j\right] && \text{(B.19)}\\
&= \frac{1}{2d}\mathbf{E}\left[(\mathbf{x}-\mathbf{x}^*)^T\nabla F(\mathbf{x})\right] && \text{(B.20)}\\
&\geq \frac{1}{2d}\mathbf{E}\left[F(\mathbf{x}) - F(\mathbf{x}^*)\right]\,. && \text{(B.21)}
\end{aligned}$$

In Eq. (B.20), we write $\frac{1}{2d}$ instead of $\frac{1}{d}$ (which Shalev-Schwartz and Tewari [2009] write), for there are another d duplicates of each of the original d weights. Eq. (B.21) holds since $F(\mathbf{x})$ is convex.

Summing over $T+1$ iterations gives:

$$\begin{aligned}
\mathbf{E}\left[\sum_{t=0}^{T}\Psi(\mathbf{x}^{(t)}) - \Psi(\mathbf{x}^{(t+1)})\right] &\geq \frac{1}{2d}\mathbf{E}\left[\sum_{t=0}^{T}F(\mathbf{x}^{(t)})\right] - \frac{T+1}{2d}F(\mathbf{x}^*) && \text{(B.22)}\\
&\geq \frac{T+1}{2d}\left[\mathbf{E}\left[F(\mathbf{x}^{(T)})\right] - F(\mathbf{x}^*)\right]\,. && \text{(B.23)}
\end{aligned}$$

Eq. (B.23) used the fact that $F(\mathbf{x}_t)$ decreases monotonically with t. Since $\sum_{t=0}^{T}\Psi(\mathbf{x}^{(t)}) - \Psi(\mathbf{x}^{(t+1)}) = \Psi(\mathbf{x}^{(0)}) - \Psi(\mathbf{x}^{(T+1)})$, rearranging the above inequality gives

$$\begin{aligned}
\mathbf{E}\left[F(\mathbf{x}_T)\right] - F(\mathbf{x}^*) &\leq \frac{2d}{T+1}\mathbf{E}\left[\Psi(\mathbf{x}^{(0)}) - \Psi(\mathbf{x}^{(T+1)})\right] && \text{(B.24)}\\
&\leq \frac{2d}{T+1}\mathbf{E}\left[\Psi(\mathbf{x}^{(0)})\right] && \text{(B.25)}\\
&= \frac{2d}{T+1}\left[\frac{\beta}{2}\|\mathbf{x}^*\|_2^2 + F(\mathbf{x}^{(0)})\right]\,, && \text{(B.26)}
\end{aligned}$$

where Eq. (B.25) used $\Psi(\mathbf{x}) \geq 0$ and Eq. (B.26) used $\mathbf{x}^{(0)} = 0$.

This bound divides by $T+1$ instead of T (which Shalev-Schwartz and Tewari [2009] do). Also, their theorem has an extra factor of $\frac{1}{2}$ on the right-hand side but should not due to the doubled length of $\mathbf{x}$.

B.1.4 Detailed Proof: Theorem 4.3.1

Start with Eq. (B.4), and note that the update rule in Eq. (4.5) implies that $\delta x_j \leq -(\nabla F(\mathbf{x}))_j$ (with $\beta = 1$ for Lasso). This gives us:

$$F(\mathbf{x} + \Delta\mathbf{x}) - F(\mathbf{x}) \quad \leq \quad -(\Delta\mathbf{x})^T(\Delta\mathbf{x}) + \tfrac{1}{2}(\Delta\mathbf{x})^T\mathbf{A}^T\mathbf{A}\Delta\mathbf{x}\,. \tag{B.27}$$

Noting that $\Delta\mathbf{x}$ can only have non-zeros in the indices in $\mathcal{P}_t$, we can rewrite this as

$$F(\mathbf{x} + \Delta\mathbf{x}) - F(\mathbf{x}) \quad \leq \quad -\sum_{j\in\mathcal{P}_t}(\delta x_j)^2 + \tfrac{1}{2}\sum_{i,j\in\mathcal{P}_t}(\mathbf{A}^T\mathbf{A})_{i,j}\delta x_i \delta x_j\,. \tag{B.28}$$

Separating out the diagonal terms in the sum over i, j and using $diag(\mathbf{A}^T\mathbf{A}) = \mathbf{1}$ gives the desired result:

$$F(\mathbf{x} + \Delta\mathbf{x}) - F(\mathbf{x}) \quad \leq \quad -\tfrac{1}{2}\sum_{j\in\mathcal{P}_t}(\delta x_j)^2 + \tfrac{1}{2}\sum_{\substack{i,j\in\mathcal{P}_t,\\ i\neq j}}(\mathbf{A}^T\mathbf{A})_{i,j}\delta x_i \delta x_j\,. \tag{B.29}$$

B.1.5 Detailed Proof: Theorem 4.3.2

This proof uses the result from Lemma 4.3.3, which is proven in detail in Sec. B.1.6.

We modify the potential function used for sequential `SCD`:

$$\Psi(\mathbf{x}) = \frac{\beta}{2}\|\mathbf{x} - \mathbf{x}^*\|_2^2 + \tfrac{1}{1-\epsilon}F(\mathbf{x}), \tag{B.30}$$

where ϵ is defined as in Eq. (B.61). Assume that $\mathtt{P}$ is chosen s.t. $\epsilon < 1$.

Write out the change in the potential function from an update $\Delta\mathbf{x}$, and rearrange terms algebraically:

$$\begin{aligned}
&\Psi(\mathbf{x}) - \Psi(\mathbf{x} + \Delta\mathbf{x}) \\
&\quad = \tfrac{\beta}{2}\left[\|\mathbf{x} - \mathbf{x}^*\|_2^2 - \|\mathbf{x} + \Delta\mathbf{x} - \mathbf{x}^*\|_2^2\right] + \tfrac{1}{1-\epsilon}\left[F(\mathbf{x}) - F(\mathbf{x} + \Delta\mathbf{x})\right] && \text{(B.31)}\\
&\quad = \tfrac{\beta}{2}\left[-2\mathbf{x}^T\Delta\mathbf{x} + 2\mathbf{x}^{*T}\Delta\mathbf{x} - (\Delta\mathbf{x})^T(\Delta\mathbf{x})\right] + \tfrac{1}{1-\epsilon}\left[F(\mathbf{x}) - F(\mathbf{x} + \Delta\mathbf{x})\right] && \text{(B.32)}\\
&\quad = \beta\left[\sum_{j\in\mathcal{P}_t} -x_j\delta x_j + x_j^*\delta x_j - \tfrac{(\delta x_j)^2}{2}\right] + \tfrac{1}{1-\epsilon}\left[F(\mathbf{x}) - F(\mathbf{x} + \Delta\mathbf{x})\right]\,. && \text{(B.33)}
\end{aligned}$$

Take the expectation w.r.t. $\mathcal{P}_t$, and use Lemma 4.3.3:

$$\begin{aligned}
&\mathbf{E}_{\mathcal{P}_t}\left[\Psi(\mathbf{x}) - \Psi(\mathbf{x} + \Delta\mathbf{x})\right] \\
&\quad = \beta\mathtt{P}\mathbf{E}_j\left[-x_j\delta x_j + x_j^*\delta x_j - \tfrac{(\delta x_j)^2}{2}\right] + \tfrac{1}{1-\epsilon}\mathbf{E}_{\mathcal{P}_t}\left[F(\mathbf{x}) - F(\mathbf{x} + \Delta\mathbf{x})\right] && \text{(B.34)}\\
&\quad \geq \beta\mathtt{P}\mathbf{E}_j\left[-x_j\delta x_j + x_j^*\delta x_j - \tfrac{(\delta x_j)^2}{2}\right] - \mathtt{P}\tfrac{1}{1-\epsilon}\mathbf{E}_j\left[\delta x_j(\nabla F(\mathbf{x}))_j + \tfrac{\beta}{2}(1+\epsilon)(\delta x_j)^2\right] && \text{(B.35)}\\
&\quad = \beta\mathtt{P}\mathbf{E}_j\left[-x_j\delta x_j + x_j^*\delta x_j - \tfrac{1}{1-\epsilon}(\delta x_j)^2\right] - \mathtt{P}\tfrac{1}{1-\epsilon}\mathbf{E}_j\left[\delta x_j(\nabla F(\mathbf{x}))_j\right] && \text{(B.36)}\\
&\quad \geq \beta\mathtt{P}\mathbf{E}_j\left[-x_j\delta x_j - x_j^*(\nabla F(\mathbf{x}))_j/\beta - \tfrac{1}{1-\epsilon}(\delta x_j)^2 - \tfrac{1}{1-\epsilon}\delta x_j(\nabla F(\mathbf{x}))_j/\beta\right]\,. && \text{(B.37)}
\end{aligned}$$

The last inequality used the update rule in Eq. (4.5), which implies $\delta x_j \geq -(\nabla F(\mathbf{x}))_j/\beta$.

Consider the two cases in the update rule in Eq. (4.5). Case 1: The update implies $\delta x_j = -x_j \geq -(\nabla F(\mathbf{x}))_j/\beta$, so

$$\begin{aligned}
&\mathbf{E}_{\mathcal{P}_t}\left[\Psi(\mathbf{x}) - \Psi(\mathbf{x}+\Delta\mathbf{x})\right] \\
&\quad \geq \beta\mathtt{P}\mathbf{E}_j\left[- x_j\delta x_j - x_j^*(\nabla F(\mathbf{x}))_j/\beta + \tfrac{1}{1-\epsilon}x_j\delta x_j + \tfrac{1}{1-\epsilon}x_j(\nabla F(\mathbf{x}))_j/\beta\right] && \text{(B.38)} \\
&\quad = \beta\mathtt{P}\mathbf{E}_j\left[\tfrac{\epsilon}{1-\epsilon}x_j\delta x_j - x_j^*(\nabla F(\mathbf{x}))_j/\beta + \tfrac{1}{1-\epsilon}x_j(\nabla F(\mathbf{x}))_j/\beta\right] && \text{(B.39)} \\
&\quad \geq \beta\mathtt{P}\mathbf{E}_j\left[- \tfrac{\epsilon}{1-\epsilon}x_j(\nabla F(\mathbf{x}))_j/\beta - x_j^*(\nabla F(\mathbf{x}))_j/\beta + \tfrac{1}{1-\epsilon}x_j(\nabla F(\mathbf{x}))_j/\beta\right] && \text{(B.40)} \\
&\quad = \mathtt{P}\mathbf{E}_j\left[(x_j - x_j^*)(\nabla F(\mathbf{x}))_j\right]. && \text{(B.41)}
\end{aligned}$$

Case 2: The update implies $\delta x_j = -(\nabla F(\mathbf{x}))_j/\beta \geq -x_j$, so

$$\begin{aligned}
\mathbf{E}_{\mathcal{P}_t}\left[\Psi(\mathbf{x}) - \Psi(\mathbf{x}+\Delta\mathbf{x})\right] &\geq \beta\mathtt{P}\mathbf{E}_j\left[- x_j\delta x_j - x_j^*(\nabla F(\mathbf{x}))_j/\beta\right] && \text{(B.42)} \\
&\geq \mathtt{P}\mathbf{E}_j\left[(x_j - x_j^*)(\nabla F(\mathbf{x}))_j\right]. && \text{(B.43)}
\end{aligned}$$

In both cases,

$$\begin{aligned}
\mathbf{E}_{\mathcal{P}_t}\left[\Psi(\mathbf{x}) - \Psi(\mathbf{x}+\Delta\mathbf{x})\right] &\geq \mathtt{P}\mathbf{E}_j\left[(x_j - x_j^*)(\nabla F(\mathbf{x}))_j\right] && \text{(B.44)} \\
&= \tfrac{\mathtt{P}}{2d}(\mathbf{x} - \mathbf{x}^*)^T\nabla F(\mathbf{x}) && \text{(B.45)} \\
&\geq \tfrac{\mathtt{P}}{2d}(F(\mathbf{x}) - F(\mathbf{x}^*)), && \text{(B.46)}
\end{aligned}$$

where the last inequality holds since $F(\mathbf{x})$ is convex.

Now sum over $T+1$ iterations (with an expectation over the $\mathcal{P}_t$ from all iterations):

$$\begin{aligned}
\mathbf{E}\left[\sum_{t=0}^{T}\Psi(\mathbf{x}^{(t)}) - \Psi(\mathbf{x}^{(t+1)})\right] &\geq \tfrac{\mathtt{P}}{2d}\mathbf{E}\left[\sum_{t=0}^{T}F(\mathbf{x}^{(t)}) - F(\mathbf{x}^*)\right] && \text{(B.47)} \\
&= \tfrac{\mathtt{P}}{2d}\left[\mathbf{E}\left[\sum_{t=0}^{T}F(\mathbf{x}^{(t)})\right] - (T+1)F(\mathbf{x}^*)\right] && \text{(B.48)} \\
&\geq \tfrac{\mathtt{P}(T+1)}{2d}\left[\mathbf{E}\left[F(\mathbf{x}^{(T)})\right] - F(\mathbf{x}^*)\right], && \text{(B.49)}
\end{aligned}$$

where Eq. (B.49) uses the result from Lemma 4.3.3, which implies that the objective is decreasing in expectation for $\mathtt{P}$ s.t. $\epsilon \leq 1$. (To see why the objective decreases in expectation, plug in the update rule in Eq. (4.5) into Eq. (B.62), and note that the right-hand side of Eq. (B.62) is negative.)

Since $\sum_{t=0}^{T}\Psi(\mathbf{x}^{(t)}) - \Psi(\mathbf{x}^{(t+1)}) = \Psi(\mathbf{x}^{(0)}) - \Psi(\mathbf{x}^{(T+1)})$, rearranging the above inequality gives

$$\begin{aligned}
\mathbf{E}\left[F(\mathbf{x}^{(T)})\right] - F(\mathbf{x}^*) &\leq \tfrac{2d}{\mathtt{P}(T+1)}\mathbf{E}\left[\Psi(\mathbf{x}^{(0)}) - \Psi(\mathbf{x}^{(T+1)})\right] && \text{(B.50)} \\
&\leq \tfrac{2d}{\mathtt{P}(T+1)}\mathbf{E}\left[\Psi(\mathbf{x}^{(0)})\right] && \text{(B.51)} \\
&= \tfrac{2d}{\mathtt{P}(T+1)}\left[\tfrac{\beta}{2}\|\mathbf{x}^*\|_2^2 + \tfrac{1}{1-\epsilon}F(\mathbf{x}^{(0)})\right]. && \text{(B.52)}
\end{aligned}$$

B.1.6 Detailed Proof: Lemma 4.3.3

Note: In this section, we assume the algorithm chooses a set of $\mathtt{P}$ distinct coordinates, not a multiset of coordinates. We mention analogous results for a multiset in Sec. B.1.7.

Starting with Assumption 4.3.1, we can rearrange terms as follows:

$$F(\mathbf{x}+\Delta\mathbf{x}) - F(\mathbf{x}) \quad \leq \quad (\Delta\mathbf{x})^T \nabla F(\mathbf{x}) + \tfrac{\beta}{2}(\Delta\mathbf{x})^T \mathbf{A}^T\mathbf{A}(\Delta\mathbf{x})\,. \tag{B.53}$$

Take the expectation w.r.t. $\mathcal{P}_t$, the set of updated weights, and use the fact that each set $\mathcal{P}_t$ is equally likely to be chosen.

$$\begin{aligned}
&\mathbf{E}_{\mathcal{P}_t}\left[F(\mathbf{x}+\Delta\mathbf{x}) - F(\mathbf{x})\right] \\
&\quad\leq \mathbf{E}_{\mathcal{P}_t}\left[\sum_{j\in\mathcal{P}_t} \delta x_j (\nabla F(\mathbf{x}))_j\right] + \tfrac{\beta}{2}\mathbf{E}_{\mathcal{P}_t}\left[\sum_{i,j\in\mathcal{P}_t} \delta x_i (\mathbf{A}^T\mathbf{A})_{i,j}\delta x_j\right] && \text{(B.54)}\\
&\quad= \mathbf{E}_{\mathcal{P}_t}\left[\sum_{j\in\mathcal{P}_t} \delta x_j (\nabla F(\mathbf{x}))_j + \tfrac{\beta}{2}(\delta x_j)^2\right] + \tfrac{\beta}{2}\mathbf{E}_{\mathcal{P}_t}\left[\sum_{\substack{i,j\in\mathcal{P}_t\\ i\neq j}} \delta x_i (\mathbf{A}^T\mathbf{A})_{i,j}\delta x_j\right] && \text{(B.55)}\\
&\quad= \mathtt{P}\mathbf{E}_j\left[\delta x_j (\nabla F(\mathbf{x}))_j + \tfrac{\beta}{2}(\delta x_j)^2\right] + \tfrac{\beta}{2}\mathtt{P}(\mathtt{P}-1)\mathbf{E}_{i,j\,:\,i\neq j}\left[\delta x_i (\mathbf{A}^T\mathbf{A})_{i,j}\delta x_j\right]\,, && \text{(B.56)}
\end{aligned}$$

where $\mathbf{E}_j\left[\right]$ denotes an expectation w.r.t. j chosen uniformly at random from $\{1,\ldots,2d\}$ and where $\mathbf{E}_{i,j\,:\,i\neq j}\left[\right]$ denotes an expectation w.r.t. a pair of distinct indices i,j chosen uniformly at random from $\{1,\ldots,2d\}$.

Since indices in $\mathcal{P}_t$ are chosen uniformly at random, the expectations may be rewritten as

$$\begin{aligned}
&\mathbf{E}_{\mathcal{P}_t}\left[F(\mathbf{x}+\Delta\mathbf{x}) - F(\mathbf{x})\right] && \text{(B.57)}\\
&\quad\leq \tfrac{\mathtt{P}}{2d}\left[(\Delta x)^T(\nabla F(\mathbf{x})) + \tfrac{\beta}{2}(\Delta x)^T(\Delta x)\right] && \text{(B.58)}\\
&\qquad+\tfrac{\beta}{2}\tfrac{\mathtt{P}(\mathtt{P}-1)}{2d(2d-1)}\left[(\Delta x)^T\mathbf{A}^T\mathbf{A}(\Delta x) - (\Delta x)^T(\Delta x)\right]\,,
\end{aligned}$$

where we are overloading the notation Δx: in Eq. (B.57), Δx only has non-zero entries in elements indexed by $\mathcal{P}_t$; in Eq. (B.58), Δx can have non-zero entries everywhere (set by the update rule in Eq. (4.5)).

The spectral radius, i.e., the largest absolute eigenvalue, of a matrix $\mathbf{M}$ may be expressed as $\max_{\mathbf{z}} \frac{\mathbf{z}^T\mathbf{M}\mathbf{z}}{\mathbf{z}^T\mathbf{z}}$; see, e.g., Gilbert [1988]. Letting ρ be the spectral radius of $\mathbf{A}^T\mathbf{A}$, upper-bound the second-order term in Eq. (B.58):

$$\begin{aligned}
&\mathbf{E}_{\mathcal{P}_t}\left[F(\mathbf{x}+\Delta\mathbf{x}) - F(\mathbf{x})\right] \\
&\quad\leq \tfrac{\mathtt{P}}{2d}\left[(\Delta x)^T(\nabla F(\mathbf{x})) + \tfrac{\beta}{2}(\Delta x)^T(\Delta x)\right] && \text{(B.59)}\\
&\qquad+\tfrac{\beta}{2}\tfrac{\mathtt{P}(\mathtt{P}-1)}{2d(2d-1)}\left[\rho(\Delta x)^T(\Delta x) - (\Delta x)^T(\Delta x)\right]\\
&\quad= \tfrac{\mathtt{P}}{2d}(\Delta x)^T(\nabla F(\mathbf{x})) && \text{(B.60)}\\
&\qquad+\tfrac{\beta}{2}\tfrac{\mathtt{P}}{2d}\left(1+\tfrac{(\mathtt{P}-1)(\rho-1)}{2d-1}\right)(\Delta x)^T(\Delta x)\,.
\end{aligned}$$

Letting

$$\epsilon = \tfrac{(\mathtt{P}-1)(\rho-1)}{2d-1}, \tag{B.61}$$

we can rewrite the right-hand side in terms of expectations over $j \in \{1, \ldots, 2d\}$ to get the lemma's result:

$$\mathbf{E}_{\mathcal{P}_t}\left[F(\mathbf{x}+\Delta\mathbf{x}) - F(\mathbf{x})\right] \quad \leq \quad \mathtt{P}\mathbf{E}_j\left[\delta x_j(\nabla F(\mathbf{x}))_j + \tfrac{\beta}{2}(1+\epsilon)(\delta x_j)^2\right]. \tag{B.62}$$

B.1.7 `Shotgun` with a Multiset

If we let the algorithm choose a multiset, rather than a set, of `P` coordinates, then we get $\epsilon = \frac{(\mathtt{P}-1)\rho}{2d}$, which gives worse scaling than the ϵ above. (Compare the two when all features are uncorrelated so that $\rho = 1$. With a set, the ϵ above indicates that we can use $\mathtt{P} = 2d$ and get good scaling; with a multiset, the changed ϵ indicates that we can be hurt by using larger `P`.) This modified analysis assumes that parallel updates of the same weight x_j will not make x_j negative. Proper write-conflict resolution can ensure this assumption holds and is viable in our multicore setting.

B.2 Details of Algorithm Runtimes in Tab. 4.1

We list details of the algorithm runtimes in Tab. 4.1, giving references back to the original works where needed. We try not to repeat elements discussed in Sec. 4.4.

B.2.1 Coordinate Descent

`Cyclic CD`: We use Theorem 16 from Saha and Tewari [2010], using the L_2 norm for Equation 5 in Definition 1: $L^2 = n^2 d$. This bound requires the isotonicity assumption. For Lasso, the isotonicity assumption is equivalent to assuming that $(\mathbf{A}^T\mathbf{A})_{ij} \leq 0$, $\forall i \neq j$.

`Stochastic CD (SCD)`: We use the results from Shalev-Schwartz and Tewari [2009], replicated in Theorem 4.2.1. We approximate $\frac{1}{n}F_{Lasso}(0) = \frac{1}{n}\|\mathbf{y}\|_2^2 \approx 1$.

`Greedy CD`: We use Lemma 1 from Dhillon et al. [2011], with $K_1 = d$.

B.2.2 Iterative Shrinkage/Thresholding

Relative to `GPSR-BB` and `SpaRSA`, the algorithm `FPC_AS` [Wen et al., 2010] requires an extra factor of d computations per iteration, for it solves subproblems using a second-order method. Note, however, that these subproblems involve only an active set of coordinates, which can be smaller than d.

B.2.3 Compressed Sensing

We do not include iteration bounds for `Hard_l0` [Blumensath and Davies, 2009] or `CoSaMP` [Needell and Tropp, 2010]. These algorithms are analyzed within the compressed sensing literature, which generally focuses on recovery (discovering a "true" $\mathbf{x}$) rather than our goal of minimizing an objective.

B.2.4 Homotopy

Our bounds for `LARS` are from Section 7 of Efron et al. [2004].

B.2.5 Stochastic Gradient

For the four `SGD` variants in Tab. 4.1, all bounds use asymptotically optimal learning rates.

`Vanilla SGD`: We use the analysis from Boyd and Mutapcic [2008]; since this analysis requires that the gradient be bounded, we omit an iteration bound for Lasso. The logistic loss iteration bound is technically w.r.t. the best iterate $\mathbf{x}^{(t)}$, not the last.

`TruncGrad`: Langford et al. [2009a] present a generalization of soft thresholding but only prove convergence bounds for the special case of soft thresholding, in their Theorem 3.2.

Since Theorem 3.2 bounds the average objective $F(\mathbf{x}^{(t)}, \cdot)$ over all iterations t, the bound we state in Tab. 4.1 is actually not for the objective at the final iterate. Also, Theorem 3.2 technically applies to modified iterates $\hat{\mathbf{x}}^{(t)}$ which are averages over all previous iterates $\mathbf{x}^{(t')}$, $t' \leq t$. Note also that their bounds use a different objective, with the loss normalized for sample size (by $1/n$); this makes no real difference, for if we redo our analysis with this modified objective, we end up with essentially the same bounds. We therefore use the same notation $F(\cdot)$ for the objective.

For logistic loss, Theorem 3.2 from Langford et al. [2009a] states:

$$\mathbf{E}\left[F(\mathbf{x}^{(T)}, \lambda)\right] \;\leq\; \tfrac{\kappa_1^2}{2}\eta + \tfrac{\|\mathbf{x}^*\|_2^2}{2\eta T} + F(\mathbf{x}^*, \lambda)\,. \tag{B.63}$$

Rearranging terms, we get:

$$\mathbf{E}\left[F(\mathbf{x}^{(T)}, \lambda)\right] - F(\mathbf{x}^*, \lambda) \;\leq\; \tfrac{\kappa_1^2}{2}\eta + \tfrac{\|\mathbf{x}^*\|_2^2}{2\eta T}\,. \tag{B.64}$$

Optimizing η gives $\eta = \frac{\|\mathbf{x}^*\|_2}{\kappa_1\sqrt{T}}$. Plugging this in gives:

$$\mathbf{E}\left[F(\mathbf{x}^{(T)}, \lambda)\right] - F(\mathbf{x}^*, \lambda) \;\leq\; \tfrac{\kappa_1\|\mathbf{x}^*\|_2}{\sqrt{T}}\,. \tag{B.65}$$

For least squares, Theorem 3.2 says:

$$\mathbf{E}\left[(1 - 2\kappa_1^2\eta)F\left(\mathbf{x}^{(T)}, \tfrac{\lambda}{1-2\kappa_1^2\eta}\right)\right] \;\leq\; \tfrac{\|\mathbf{x}^*\|_2^2}{2\eta T} + F(\mathbf{x}^*, \lambda)\,, \tag{B.66}$$

Replace λ with $(1 - 2\kappa_1^2\eta)\lambda$:

$$\begin{aligned} \mathbf{E}\left[(1 - 2\kappa_1^2\eta)F(\mathbf{x}^{(T)}, \lambda)\right] \;&\leq\; \tfrac{\|\mathbf{x}^*\|_2^2}{2\eta T} + F\left(\mathbf{x}^*, (1 - 2\kappa_1^2\eta)\lambda\right) && \text{(B.67)} \\ &=\; \tfrac{\|\mathbf{x}^*\|_2^2}{2\eta T} + F\left(\mathbf{x}^*, \lambda\right) - 2\kappa_1^2\eta\|\mathbf{x}^*\|_1\,. && \text{(B.68)} \end{aligned}$$

Multiply both sides by $1/(1 - 2\kappa_1^2\eta)$, and rearrange terms:

$$\mathbf{E}\left[F(\mathbf{x}^{(T)}, \lambda)\right] \;\leq\; \tfrac{1}{1-2\kappa_1^2\eta}\tfrac{\|\mathbf{x}^*\|_2^2}{2\eta T} + \tfrac{1}{1-2\kappa_1^2\eta}F\left(\mathbf{x}^*, \lambda\right) - \tfrac{2\kappa_1^2\eta}{1-2\kappa_1^2\eta}\|\mathbf{x}^*\|_1 \tag{B.69}$$

$$\begin{aligned}
\mathbf{E}\left[F(\mathbf{x}^{(T)},\lambda)\right] - F(\mathbf{x}^*,\lambda) &\leq \tfrac{1}{1-2\kappa_1^2\eta}\tfrac{\|\mathbf{x}^*\|_2^2}{2\eta T} + \tfrac{2\kappa_1^2\eta}{1-2\kappa_1^2\eta}F(\mathbf{x}^*,\lambda) - \tfrac{2\kappa_1^2\eta}{1-2\kappa_1^2\eta}\|\mathbf{x}^*\|_1 && \text{(B.70)}\\
&= \tfrac{1}{1-2\kappa_1^2\eta}\tfrac{\|\mathbf{x}^*\|_2^2}{2\eta T} + \tfrac{2\kappa_1^2\eta}{1-2\kappa_1^2\eta}F(\mathbf{x}^*,\lambda-1) && \text{(B.71)}\\
&= \tfrac{1}{1-2\kappa_1^2\eta}\left(\tfrac{\|\mathbf{x}^*\|_2^2}{2\eta T} + 2\kappa_1^2\eta F(\mathbf{x}^*,\lambda-1)\right) && \text{(B.72)}
\end{aligned}$$

If we ignore the leading $\frac{1}{1-2\kappa_1^2\eta}$ factor, optimizing η gives us $\eta = \frac{1}{2}\frac{\|\mathbf{x}^*\|_2}{\kappa_1\sqrt{T}}$, which is $1/2$ times the optimal η for logistic loss. Plugging this in, we get:

$$\begin{aligned}
\mathbf{E}\left[F(\mathbf{x}^{(T)},\lambda)\right] - F(\mathbf{x}^*,\lambda) &\leq \tfrac{1}{1-\kappa_1\frac{\|\mathbf{x}^*\|_2}{\sqrt{T}}}\tfrac{\|\mathbf{x}^*\|_2\kappa_1}{\sqrt{T}} + \tfrac{\kappa_1\frac{\|\mathbf{x}^*\|_2}{\sqrt{T}}}{1-\kappa_1\frac{\|\mathbf{x}^*\|_2}{\sqrt{T}}}F(\mathbf{x}^*,\lambda-1) && \text{(B.73)}\\
&= \tfrac{1}{1-\frac{\kappa_1\|\mathbf{x}^*\|_2}{\sqrt{T}}}\tfrac{\kappa_1\|\mathbf{x}^*\|_2}{\sqrt{T}}\left(1+F(\mathbf{x}^*,\lambda-1)\right) && \text{(B.74)}\\
&\approx \tfrac{\kappa_1\|\mathbf{x}^*\|_2}{\sqrt{T}}\left(1+F(\mathbf{x}^*,\lambda-1)\right)\,, && \text{(B.75)}
\end{aligned}$$

where the last approximate equality assumes that T is large enough that $\frac{\kappa_1\|\mathbf{x}^*\|_2}{\sqrt{T}}$ is small. In Tab. 4.1, we ignore the $(1+F(\mathbf{x}^*,\lambda-1))$ term, which is relatively small since the loss is normalized by $1/n$.

`SMIDAS`: We use Theorem 2 from Shalev-Schwartz and Tewari [2009].

`RDA`: The Lasso iteration bound is for the constraint formulation in Eq. (4.12). Their analysis technically does not apply to the penalty formulation in Eq. (4.2), for Theorem 2 in Xiao [2010] requires that the gradient be bounded on the feasible region. In their Theorem 2, we use $L^2 = c^2n^2d$ for Lasso and $L^2 = n^2\kappa_1$ for logistic regression. `RDA` takes d operations per iteration, for it maintains a sum of all gradients, which will be dense in general.

Scaling w.r.t. dimensionality d: The `SGD` bounds all include additional factors which can increase as $O(d)$. The value $\kappa_1 \doteq \max_i\|\mathbf{a}_i\|_2^2$ can increase linearly in d if the average element magnitude of $\mathbf{a}_i$ remains constant. The bound $\|\mathbf{x}^*\|_1^2 \leq \|\mathbf{x}^*\|_0\|\mathbf{x}^*\|_2^2$ can be tight, and $\|\mathbf{x}^*\|_0$ could grow linearly in d, so the $\|\mathbf{x}^*\|_1^2$ factor in `SMIDAS`'s bound could be replaced with $d\|\mathbf{x}^*\|_2^2$ in the worst case.

B.2.6 Accelerated

`FISTA`: The bound in Tab. 4.1 uses Theorem 4.4 from Beck and Teboulle [2009], in which we let $\alpha = 1$ (for constant stepsize), we use the L_2 norm for $\|\mathbf{x}^*\|$, and the Lipschitz constant is $L = \rho$ for both Lasso and logistic regression. Each iteration takes an unknown number of sub-iterations k for a backtracking line search.

B.2.7 Interior Point

Since Kim et al. [2007] and Koh et al. [2007] do not provide convergence analysis, we use the analysis from Boyd and Vandenberghe [2004] for iteration bounds (the number of centering steps). In these bounds, ε is the duality gap, which is related to but not identical to the distance from the optimum. The iteration bound is on the total number of inner iterations (conjugate gradient steps) within each outer iteration (centering step). The cost per iteration is the cost of one conjugate gradient step. Both methods phrase L_1 regularization in constraint form using $2d$ constraints.

L1_LS and L1_logreg: We use the iteration bound from Section 11.3.3 from Boyd and Vandenberghe [2004]. The number k of inner iterations (of conjugate gradient) does not appear in the iteration cost, for the $knds$ cost of conjugate gradient is outweighed by the nd^2 cost of computing the Hessian.

We approximate $\kappa_3 \approx \sqrt{d}$ based on the iteration bound from Section 11.5.3 from Boyd and Vandenberghe [2004]. This bound is technically applicable to squared error but not logistic loss. Also, this bound is based on Newton steps instead of conjugate gradient, so this approximation is an underestimate.

The Hessian computation prevents this method from scaling with the sparsity of $\mathbf{A}$. The conjugate gradient steps do scale with the sparsity of $\mathbf{A}$ but are outweighted by the Hessian computation.

B.2.8 Distributed

ADMM: See Sec. 4.4.5.

DDA: The convergence bounds we state are for expander graphs (item (d) in Corollary 1 of Theorem 2 from Duchi et al. [2012]), which have the best convergence rate of the graphs they consider. Using the proximal function $\Psi(x) = \frac{1}{2}\|\mathbf{x}\|_2^2$, we get $R = \frac{1}{\sqrt{2}}\|\mathbf{x}^*\|_2$. We use the constraint formulation since their proofs require that the loss be Lipschitz on the feasible region. For squared error, $L = c\rho$. For logistic loss, $L = n\sqrt{d}\max|A_{ij}| \approx n\sqrt{d}$.

Bibliography

Pieter Abbeel, Daphne Koller, and Andrew Y. Ng. Learning factor graphs in polynomial time and sample complexity. *J. Mach. Learn. Res.*, 7:1743–1788, December 2006. ISSN 1532-4435. URL `http://portal.acm.org/citation.cfm?id=1248547.1248611`. 1.2, 1.5.1, 2, 2.1.1, 2.1.3, 2.6, 2.6.1, 2.6.1, 2.6.1, 2.6.1, 2.6.1, 2.6.3, 2.6.4, 2.6.1, 2.6.6, 2.6.1, 2.6.7, 2.6.1, 2.6.1, 2.6.2, 2.6.4, 2.6.7, 2.6.7, 2.8.2, 5

Animashree Anandkumar, Dean P. Foster, Daniel Hsu, Sham M. Kakade, and Yi-Kai Liu. Two svds suffice: Spectral decompositions for probabilistic topic modeling and latent dirichlet allocation. In *NIPS*, 2012. 5.3.1

A.U. Asuncion, Q. Liu, A.T. Ihler, and P. Smyth. Learning with blocks: Composite likelihood and contrastive divergence. In *Artificial Intelligence and Statistics (AISTATS)*, 2010. 2.1.2, 2.8.1, 5.3.1

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970. 5.3.1, 5.3.2

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009. 4.4.3, B.2.6

J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195, 1975. 1.2, 2, 2.1.1, 2.2

T. Blumensath and M.E. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3), 2009. 4.4.4, 4.5.1, B.2.3

Leon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *NIPS*, 2007. 4.4.2

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 0521833787. B.2.7

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. In *Foundations and Trends in Machine Learning*. Now Publishers, 2011a. 2.8.1, 4.7.4, 6.2

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. In *Foundations and Trends in Machine Learning*. Now Publishers, 2011b. 4.4.5

Stephen Boyd and Almir Mutapcic. Stochastic subgradient methods. Lecture Notes for EE364b, Stanford University, 2008. `http://see.stanford.edu/materials/lsocoee364b/04-stoch_subgrad_notes.pdf`. B.2.5

Joseph Bradley and Carlos Guestrin. Learning tree conditional random fields. In *Proc. of the 27th Annual International Conference on Machine Learning*, 2010. 3, 5.3.3

Joseph K. Bradley and Carlos Guestrin. Sample complexity of composite likelihood. In *AISTATS*, 2012. 2, 2.6.5, 6.1

Joseph K. Bradley, Aapo Kyrola, Danny Bickson, and Carlos Guestrin. Parallel coordinate descent for l_1-regularized loss minimization. In *Proc. of the 28th Annual International Conference on Machine Learning*, 2011. 1.4, 4, 4.1

Alfred M. Bruckstein, David L. Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, 2009. 4.1

Lucien Le Cam. *Asymptotic Methods in Statistical Decision Theory*. Springer, 1986. 5.3.1, 6.1

E.J. Candes and T. Tao. Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203–4215, 2005. 4.3.3

Emmanuel J. Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. on Information Theory*, 52(12):5406–5425, 2006. 4.3.3

Andrew Carlson. *Coupled Semi-Supervised Learning*. PhD thesis, Carnegie Mellon University, 2010. 6.4.2

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*, 2010. 6.4, 6.4.1, 6.4.2

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 5

Anton Chechetka and Carlos Guestrin. Efficient principled learning of thin junction trees. In *NIPS*, 2007. 1.3

Anton Chechetka and Carlos Guestrin. Evidence-specific structures for rich tractable crfs. In *In Advances in Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2010. 3.1, 5.3.3

C.K. Chow and C.N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Info. Theory*, 14:462–467, 1968. 1, 1.3, 1.3, 1.5.1, 3, 3.1, 3.2.1, 5, 5.3.3, 5.3.4

Adam Coates, Paul Baumstarck, Quoc Le, and Andrew Y. Ng. Scalable learning for object detection with gpu hardware. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009. 4.7.3

Adnan Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM*, 50(3):280–305, 2003. 5.3.1

J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1), 2004. 6.2

Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13:165–202, 2012. 4.4.2

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977. 5.3.1, 5.3.2

Inderjit S. Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Nearest neighbor based greedy coordinate descent. In *NIPS*, 2011. 4.4.1, B.2.1

J. Dillon and G. Lebanon. Stochastic composite likelihood. *JMLR*, 11:2597–2633, 2010. 2.1.2

M.F. Duarte, M.A. Davenport, D. Takhar, J.N. Laska, T. Sun, K.F. Kelly, and R.G. Baraniuk. Single-pixel imaging via compressive sampling. *Signal Processing Magazine, IEEE*, 25(2), 2008. 4.3.2, 4.5.1

John Duchi, Alekh Agarwal, and Martin Wainwright. Dual averaging for distributed optimization: Convergence and network scaling. *To appear in IEEE Transactions on Automatic Control*, 2012. 4.4.5, B.2.8

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of statistics*, 32(2), 2004. 4.4.1, 4.5.1, B.2.4

J. Eidsvik, B.A. Shaby, B.J. Reich, M. Wheeler, and J. Niemi. Estimation and prediction in spatial models with block composite likelihoods using parallel computing. Technical report, NTNU, Duke, NCSU, UCSB, under submission. 2.3.4

Pedro F. Felzenszwalb and Julian J. McAuley. Fast inference with min-sum matrix product. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2549–2554, 2011. 5.3.1

M.A.T Figueiredo, R.D. Nowak, and S.J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *Selected Topics in Signal Processing, IEEE Journal of*, 2008. 4.4, 4.5.1

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1), 2010. 4.5.1, 4.5.1

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008. 4.7.2

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997. 3.1, 3.2.1

W.J. Fu. Penalized regressions: The bridge versus the lasso. *Journal of Comp. and Graphical Statistics*, 1998. 4.1

S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984. 2.1.1, 5.3.1

B. Gidas. Consistency of maximum likelihood and pseudolikelihood estimators for gibbs distributions. In *Proc. of Workshop on Stochastic Differential Systems with Applications in Electrical/Computer Engineering, Control Theory, and Operations Research*, 1986. 2.1.2

S. Gilbert. *Linear Algebra and Its Applications*. Harcourt Brace Jovanovich, 3rd edition, 1988. 4.3.1, B.1.6

Joseph Gonzalez, Yucheng Low, and Carlos Guestrin. Residual splash for optimally parallelizing belief propagation. In *In Artificial Intelligence and Statistics (AISTATS)*, Clearwater Beach, Florida, April 2009. 5.3.1

Joseph E. Gonzalez, Yucheng Low, Arthur Gretton, and Carlos Guestrin. Parallel gibbs sampling: From colored fields to thin junction trees. In *AISTATS*, 2011. 5.2, 5.3.2

Henry W. Gould. Combinatorial identities: Table i: Intermediate techniques for summing finite series, vol. 4. `http://www.math.wvu.edu/~gould/Vol.4.PDF`, 2010. A.2.2

Scott Grauer-Gray, Chandra Kambhamettu, and Kannappan Palaniappan. Gpu implementation of belief propagation using cuda for cloud tracking and reconstruction. In *IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS 2008)*, 2008. 4.7.3

Bingsheng He and Xiaoming Yuan. On the o(1/n) convergence rate of douglas-rachford alternating direction method. *Appearing in SIAM J. on Numerical Analysis*, 2012. `http://www.math.hkbu.edu.hk/~xmyuan/Paper/HeYuan-ADM-Complexity.pdf`. 4.4.5

G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14:1771–1800, 2002. 1.2, 2.1.1, 5.3.1, 6.1

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.*, 58:13–30, 1963. A.1.2, A.1.2, A.1.3

K.-U. Höffgen. Learning and robust learning of product distributions. In *COLT*, 1993. 3.3.4

Matt Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. Arxiv 1206.7051v1, 2012. 5.3.1

R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 1990. A.1.2

A. Hyvärinen. Estimation of non-normalized statistical models by score matching. *JMLR*, 6:695–709, 2005. 2.1.1, 2.8.2, 5.3.1, 6.1

A. Hyvärinen. Consistency of pseudolikelihood estimation of fully visible boltzmann machines. *Neural Computation*, 18(10): 2283–2292, 2006. 2.1.2

A. Hyvärinen. Some extensions of score matching. *Computational Statistics & Data Analysis*, 51(5):2499–2512, 2007. 2.1.1, 2.8.2, 5.3.1, 6.1

J. Gillenwater K. Ganchev, J. Graa and B. Taskar. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research (JMLR)*, 2010. 2.8.3, 5.3.1, 5.3.2

S. J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale ℓ1-regularized least squares. *IEEE Journal on Selected Topics in Signal Processing*, 1(4):606–617, 2007. 4.1, 4.4.1, 4.4.3, 4.5.1, B.2.7

Ross Kindermann and James Laurie Snell. *Markov Random Fields and Their Applications*. American Mathematical Society, 1980. 1.1

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983. 3.6.2

S. Kogan, D. Levin, B.R. Routledge, J.S. Sagi, and N.A. Smith. Predicting risk from financial reports with regression. In *Human Language Tech.-NAACL*, 2009. 4.5.1, 4.5.1

K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *JMLR*, 8:1519–1555, 2007. 4.4.1, 4.4.3, 4.5.2, B.2.7

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009. 1, 1.1

J.B. Kruskal, Jr. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. AMS*, 7(1):48–50, 1956. 3.2.2

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001. (document), 1, 1.1, 1.1.3, 1.2, 2.1.1

John Lafferty, Yan Liu, and Xiaojin Zhu. Kernel conditional random fields: Representation, clique selection, and semi-supervised learning. In *ICML*, 2004. 2.8.3

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. In *NIPS*, 2009a. 4.4.2, 4.5.2, B.2.5

J. Langford, A.J. Smola, and M. Zinkevich. Slow learners are fast. In *NIPS*, 2009b. 4.1

Quoc V. Le, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, and Andrew Y. Ng. On optimization methods for deep learning. In *ICML*, 2011. 5.3.1, 6.1

Su-In Lee, Varun Ganapahthi, and Daphne Koller. Efficient structure learning of markov networks using l1-regularization. In *NIPS*, 2006. 3.1

C. E. Leiserson. The Cilk++ concurrency platform. In *46th Annual Design Automation Conference*. ACM, 2009. 4.5.1

D.D. Lewis, Y. Yang, T.G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 5, 2004. 4.5.2

Percy Liang and Michael Jordan. An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *Proc. of the 25th Annual International Conference on Machine Learning*, 2008. 2, 2.1.1, 2.1.2, 2.3.1, 2.3.1, 2.3.3, 2.8.2, 5.3.2, 6.1

B. Lindsay. Composite likelihood methods. *Contemporary Mathematics*, 80:221–239, 1988. 1.2, 2, 2.1.1, 2.2

Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1–3):503–528, 1989. 5.3.1

Han Liu, Kathryn Roeder, and Larry Wasserman. Stability approach to regularization selection (stars) for high dimensional graphical models. In *NIPS*, 2010a. 5.3.4

Han Liu, Min Xu, Haijie Gu, Anupam Dasgupta, John Lafferty, and Larry Wasserman. Forest density estimation. In *Proceedings of the 23rd Annual Conference on Learning Theory*, 2010b. 3.6.2

Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. Graphlab: A new parallel framework for machine learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, California, July 2010. 5.1, 6.2

Daniel Lowd. Closed-form learning of markov networks from dependency networks. In *UAI*, 2012. 2.1.2

Daniel Lowd and Pedro Domingos. Learning arithmetic circuits. In *UAI*, 2008. 5.3.1, 5.3.4

G. Mann, R. McDonald, M. Mohri, N. Silberman, and D. Walker. Efficient large-scale distributed training of conditional maximum entropy models. In *NIPS*, 2009. 4.1

Benjamin M. Marlin and Nando de Freitas. Asymptotic efficiency of deterministic estimators for discrete energy-based models: Ratio matching and pseudolikelihood. 6.1

Andrew McCallum, Gideon Mann, and Gregory Druck. Generalized expectation criteria. Technical Report 2007-60, U. of Massachusetts Amherst, 2007. 2.8.3, 5.3.1, 5.3.2

N. Meinshausen and B. Yu. Lasso-type recovery of sparse representations for high-dimensional data. *Annals of Statistics*, 37(1): 246–270, 2009. 4.3.3

K. Nakazatoa and T. Arita. A growth model of community graph with a degree distribution consisting of two distinct parts. *Physica A: Stat. Mech. and Apps.*, 376:673–678, 2007. 3.4.1

Deanna Needell and Joel A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Communications of the ACM*, 53(12):93–100, 2010. 4.4.4, B.2.3

Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. CORE Discussion Papers 2010002, January 2010. 4.2.2, 4.4.3

A.Y. Ng. Feature selection, l_1 vs. l_2 regularization and rotational invariance. In *ICML*, 2004. 1.4, 4.1, 4.2

Feng Niu, Benjamin Recht, Christopher Re, and Stephen J. Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, 2011. 4.4.5

M. Palatucci, D. Pomerleau, G. Hinton, and T. Mitchell. Zero-shot learning with semantic output codes. In *NIPS*, 2009. 3.4.2

N. Parikh and S. Boyd. Graph projection block splitting for distributed optimization. submitted, 2012. `http://www.stanford.edu/~boyd/papers/block_splitting.html`. 4.4.5, 4.7.4

Judea Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proc. of the 7th Conference of the Cognitive Science Society*, pages 329–334, 1985. 1.1

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2nd edition, 1988. 5.3.1

Hoifung Poon and Pedro Domingos. Machine reading: A k̈iller app̈for statistical relational ai. In *AAAI*, 2010. 6.4

Rajat Raina, Anand Madhavan, and Andrew Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *Proc. of the 26th Annual International Conference on Machine Learning*, 2009. 4.7.3

P. Ravikumar, M. J. Wainwright, G. Raskutti, and B. Yu. Model selection in gaussian graphical models: High-dimensional consistency of l1-regularized mle. In *NIPS*, 2008. 1.3, 1.4, 3.2, 3.3.5

Pradeep Ravikumar, Martin J. Wainwright, and John Lafferty. High-dimensional ising model selection using l1-regularized logistic regression. *Annals of Statistics*, 38(3):1287–1319, 2010. 1.3, 1.4, 2.1.2, 2.3.1, 3.1, 3.6.2, 3.6.3, 5.3.3, 5.3.4

M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006. 6.4

Saharon Rosset and Eran Segal. Boosting density estimation. In *In Advances in Neural Information Processing Systems 15*, pages 641–648. MIT Press, 2002. 3.6.1

Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82:273–302, 1996. 1.1.2

Sushmita Roy, Terran Lane, and Margaret Werner-Washburne. Learning structurally consistent undirected probabilistic graphical models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 905–912, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: http://doi.acm.org/10.1145/1553374.1553490. URL `http://doi.acm.org/10.1145/1553374.1553490`. 2.1.3, 2.6.1, 2.1, 2.6.1, 2.6.2, 2.6.1, 2.6.3, 2.6.4, 2.6.6, 2.6.7, 2.6.1, 2.6.2, 2.6.3, 2.6.4

Ankan Saha and Ambuj Tewari. On the finite time convergence of cyclic coordinate descent methods, 2010. preprint arXiv:1005.2146. 4.4.1, B.2.1

Chad Scherrer, Mahantesh Halappanavar, Ambuj Tewari, and David Haglin. Scaling up coordinate descent algorithms for large ℓ_1 regularization problems. In *ICML*, 2012a. 4.4.5, 4.7.1, 5.2

Chad Scherrer, Ambuj Tewari, Mahantesh Halappanavar, and David Haglin. Feature clustering for accelerating parallel coordinate descent. In *NIPS*, 2012b. 4.4.5, 4.7.1, 5.2

M. Schmidt, K. Murphy, G. Fung, and R. Rosales. Structure learning in random fields for heart motion abnormality detection. In *CVPR*, 2008. 1, 1.1, 1.3, 2.1.1, 2.1.2, 3.1, 3.2, 3.2.2, 3.4, 3.4.1, 3.6.2, 3.6.3, 5.3.4

D. Shahaf, A. Chechetka, and C. Guestrin. Learning thin junction trees via graph cuts. In *AI-Stats*, 2009. 1.3, 3, 3.1, 3.2, 3.2.1, 3.2.2, 3.6.3, 5.3.3, 5.3.4

Shai Shalev-Schwartz and Ambuj Tewari. Stochastic methods for ℓ_1 regularized loss minimization. In *ICML*, 2009. 4.1, 4.2, 4.2.1, 4.2.1, 4.2.1, 4.2.1, 4.2.2, 4.4.1, 4.4.2, 4.5.2, B.1.2, B.1.3, B.1.3, B.2.1, B.2.5

S. Shalev-Shwartz and A. Tewari. Stochastic methods for ℓ_1 regularized loss minimization. *JMLR*, (12):1865–1892, 2011. 4.2.1

Le Song, Jonathan Huang, Alex Smola, and Kenji Fukumizu. Hilbert space embeddings of conditional distributions. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009. 2.8.3

Le Song, Byron Boots, Sajid Siddiqi, Geoffrey Gordon, and Alex Smola. Hilbert space embeddings of hilbert markov models. In *Proceedings of the 27th Annual International Conference on Machine Learning*, 2010. 2.8.3

N. Srebro. Maximum likelihood bounded tree-width markov networks. *AI*, 143(1):123–138, 2003. 1.3, 3.1

C. Sutton and A. McCallum. Piecewise training for undirected models. In *UAI*, 2005. 1.2, 2.1.1, 2.1.2, 2.8.2, 3.1, 3.3.1, 3.3.1, 5.3.1, 6.1

C. Sutton and A. McCallum. Piecewise pseudolikelihood for efficient training of conditional random fields. In *ICML*, 2007. 2.1.1, 2.1.2, 2.8.2, 3.3.1, 3.3.1

M.F. Tappen, C. Liu, E.H. Adelson, and W.T. Freeman. Learning gaussian conditional random fields for low-level vision. In *CVPR*, 2007. 3.4.2

Marc Teyssier and Daphne Koller. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *Proc. of the 21st Conference on Uncertainty in AI (UAI)*, 2005. 1.3, 5.3.3, 5.3.4

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal Statistical Society*, 58(1):267–288, 1996. 4.1, 4.2

A. Torralba, K. Murphy, and W. Freeman. Contextual models for object detection using boosted random fields. In *NIPS*, pages 1401–1408, 2004. 1.3, 3.1, 5.3.3, 5.3.4

J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. AC-31(9), 1986. 4.4.5

D. L. Vail, M. M. Veloso, and J. D. Lafferty. Conditional random fields for activity recognition. In *Proc. 6th Intl. Joint Conf. on Autonomous Agents and Multiagent Systems*, pages 1–8, 2007. 1, 1.1

L.G. Valiant. A theory of the learnable. *Comm. ACM*, 27(11):1134–1142, 1984. 1.2, 2

E. van den Berg, M.P. Friedlander, G. Hennenfent, F. Herrmann, R. Saab, and O. Yılmaz. Sparco: A testing framework for sparse reconstruction. *ACM Transactions on Mathematical Software*, 35(4), 2009. 4.3.2, 4.5.1

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 5.2

S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *ICML*, 2006. 2.1.1, 6.1

M. Wainwright. Estimating the "wrong" graphical model: Benefits in the computation-limited setting. *Journal of Machine Learning Research*, 7:1829–1859, 2006. 2.1.1, 2.8.1, 5.3.2, 6.1

M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions* *on Information Theory*, 51(7):2313–2335, 2005. 5.3.1

Huahua Wang and Arindam Banerjee. Online alternating direction method. In *ICML*, 2012. 4.4.5

D. Weiss and B. Taskar. Structured prediction cascades. In *AISTATS*, 2010. 5.2

Z. Wen, D. Yin, W. Goldfarb, and Y. Zhang. A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization and continuation. *SIAM Journal on Scientific Computing*, 32(4), 2010. 4.1, 4.4.4, 4.5.1, B.2.2

Michael Wick, Andrew McCallum, and Gerome Miklau. Scalable probabilistic databases with factor graphs and mcmc. In *VLDB*, 2010. 6.4, 6.4.2

Michael Wick, Khashayar Rohanimanesh, Kedar Bellare, Aron Culotta, and Andrew Mccallum. Samplerank: Training factor graphs with atomic gradients. In *ICML*, 2011. 5.3.1

S.J. Wright, D.R. Nowak, and M.A.T. Figueiredo. Sparse reconstruction by separable approximation. *Signal Processing, IEEE Transactions on*, 57(7), 2009. 4.1, 4.5.1

W.A. Wulf and S.A. McKee. Hitting the memory wall: Implications of the obvious. *ACM SIGARCH Computer Architecture News*, 23(1), 1995. 4.5.3

Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *JMLR*, 11:2543–2596, 2010. 4.4.2, B.2.5

A. Yang, A. Ganesh, Z. Zhou, S. Sastry, and Y. Ma. A review of fast ℓ_1-minimization algorithms for robust face recognition. *preprint arXiv:1007.3753*, 2010. 4.4

G. X. Yuan, K. W. Chang, C. J. Hsieh, and C. J. Lin. A comparison of optimization methods and software for large-scale l1-reg. linear classification. *JMLR*, 11, 2010. 4.1, 4.2.2, 4.4, 4.5.2, 4.5.2, 4.5.2

Yuchen Zhang, John C. Duchi, and Martin Wainwright. Communication-efcient algorithms for statistical optimization. In *NIPS*, 2012. 4.7.4, 5.1, 6.2

M. Zinkevich, M. Weimer, A.J. Smola, and L. Li. Parallelized stochastic gradient descent. In *NIPS*, 2010. 4.1, 4.4.2, 4.5.2, 4.7.4, 5.1, 6.2