

ABSTRACT

Title of dissertation: HINGE-LOSS MARKOV RANDOM FIELDS
AND PROBABILISTIC SOFT LOGIC:
A SCALABLE APPROACH
TO STRUCTURED PREDICTION

Stephen Hilliard Bach,
Doctor of Philosophy, 2015

Dissertation directed by: Professor Lise Getoor
Department of Computer Science

A fundamental challenge in developing impactful artificial intelligence technologies is balancing the ability to model rich, structured domains with the ability to scale to big data. Many important problem areas are both richly structured and large scale, from social and biological networks, to knowledge graphs and the Web, to images, video, and natural language. In this thesis I introduce two new formalisms for modeling structured data, distinguished from previous approaches by their ability to both capture rich structure and scale to big data. The first, hinge-loss Markov random fields (HL-MRFs), is a new kind of probabilistic graphical model that generalizes different approaches to convex inference. I unite three views of inference from the randomized algorithms, probabilistic graphical models, and fuzzy logic communities, showing that all three views lead to the same inference objective. I then derive HL-MRFs by generalizing this unified objective. The second new formalism, probabilistic soft logic (PSL), is a probabilistic programming language that makes HL-MRFs easy to define, refine, and reuse for relational data. PSL uses a syntax based on first-order logic to compactly specify complex models. I next introduce an algorithm for inferring most-probable variable assignments (MAP inference) for HL-MRFs that is extremely scalable, much more so than commercially available software, because it uses message passing to leverage the sparse dependency structures common in inference tasks. I then show how to learn the parameters of HL-MRFs using a number of learning objectives. The learned HL-MRFs are as accurate as traditional, discrete models, but much more scalable. To enable HL-MRFs and PSL to capture even richer dependencies, I then extend learning to support latent variables, i.e., variables without training labels. To overcome the bottleneck of repeated inferences required during learning, I introduce paired-dual learning, which interleaves inference and parameter updates. Paired-dual learning learns accurate models and is also scalable, often completing before traditional methods make even one parameter update. Together, these algorithms enable HL-MRFs and PSL to model rich, structured data at scales not previously possible.

HINGE-LOSS MARKOV RANDOM FIELDS
AND PROBABILISTIC SOFT LOGIC:
A SCALABLE APPROACH TO STRUCTURED PREDICTION

by

Stephen Hilliard Bach

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Advisory Committee:
Professor Lise Getoor, Chair/Advisor
Professor Rama Chellappa
Professor Hal Daumé III
Professor Larry S. Davis
Dr. Kevin Murphy

© Copyright by
Stephen Hilliard Bach
2015

This thesis is dedicated to Alexis,
for all her love, encouragement, and support.

Acknowledgments

I first and foremost owe a great debt of gratitude to my advisor, Lise Getoor, who shaped me into a researcher with her great wisdom, patience, and kindness. Over the past five years, she has always had my best interests at heart, which gave me the confidence to navigate the ups and downs of graduate school. Lise taught me to be curious yet disciplined and how to share my ideas with the research community. I am so grateful to Lise for believing in me and supporting me.

Bert Huang also significantly shaped my development as a researcher. During his time as a postdoctoral research associate at the University of Maryland, I was fortunate to collaborate with him on many projects. He was an equal contributor to the development of paired-dual learning. From marathon whiteboard sessions to working together on experiments late into the night, Bert inspired me to persevere when the research was hard and to do the best work I could. Bert is also a friend and a guide who has helped me grow in my career. I will miss working regularly with him.

I also want to thank Matthias Broecheler. Matthias took me under his wing when I first began graduate school, showing me how to turn questions into research ideas and research ideas into contributions. He was patient and generous with his time as I learned to find my own way. Matthias paved the way for probabilistic soft logic by developing probabilistic similarity logic and helping me as I began to work on hinge-loss Markov random fields. I am very grateful for all he has taught me.

During my time at the University of Maryland, faculty members guided me generously. Dianne O’Leary taught me about convex optimization and advised me during the development of MAP inference for hinge-loss Markov random fields. Jordan Boyd-Graber was a guide during the development of paired-dual learning. Thank you both for being great teachers.

I entered Lise’s group, LINQS, at the same time as three other students: Ben London, Jay Pujara, and Theodoros Rekatsinas. Since then, they have become friends, collaborators, and emotional supports. Thanks guys, for traveling down this path with me. Thank you also to the rest of the LINQS group, past and present. I have been fortunate to work with many great students and postdocs.

Someone very special to me, Alexis, supported me immensely during the writing of this thesis. Thank you for always encouraging me.

Finally, I want to thank my mom and dad, Lisa and Al, for all their love. They have made so many sacrifices for my education and my well-being. Words cannot fully express how grateful I am to them.

Portions of this work were supported by NSF grants CCF0937094 and IIS1218488, and IARPA via DoI/NBC contract number D12PC00337. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

Table of Contents

1	Introduction	1
2	Related Work	14
2.1	Models and Languages	17
2.2	Inference	19
2.3	Learning	23
3	Hinge-Loss Markov Random Fields	26
3.1	Unifying Inference in Logic-Based Models	26
3.1.1	MAX SAT Relaxation	31
3.1.2	Local Consistency Relaxation	33
3.1.3	Łukasiewicz Logic	36
3.2	Generalizing Convex Inference	37
3.2.1	Relaxed Linear Constraints	39
3.2.2	Hard Linear Constraints	40
3.2.3	Generalized Hinge-Loss Functions	41
3.3	Definition	43
4	Probabilistic Soft Logic	45
4.1	Definition	46
4.1.1	Preliminaries	46
4.1.2	Inputs	49
4.1.3	Rules and Grounding	53
4.1.4	Logical Rules	53
4.1.5	Arithmetic Rules	58
4.2	Expressivity	64
4.3	Modeling Patterns	68
4.3.1	Domain and Range Rules	68
4.3.2	Similarity	70
4.3.3	Priors	70
4.3.4	Blocks and Canopies	71
4.3.5	Aggregates	72

5	MAP Inference	76
5.1	Consensus Optimization Formulation	77
5.2	Block Updates	80
5.3	Lazy MAP Inference	85
5.4	Evaluation of MAP Inference	86
5.4.1	Scalability	87
5.4.2	Accuracy for Approximate Discrete Inference	91
6	Supervised Weight Learning	97
6.1	Maximum Likelihood Estimation and Structured Perceptron	98
6.2	Maximum Pseudolikelihood Estimation	99
6.3	Large-Margin Estimation	100
6.4	Evaluation of Supervised Learning	103
6.4.1	Node Labeling	104
6.4.2	Link Labeling	106
6.4.3	Link Prediction	109
6.4.4	Image Completion	113
7	Learning with Latent Variables	116
7.1	Background on Variational Expectation Maximization	118
7.2	Paired-Dual Learning	120
7.2.1	Tractable Entropy Surrogates	121
7.2.2	Joint Optimization	123
7.2.3	Learning Algorithm	125
7.3	Related Approaches for Discrete Models	128
7.4	Evaluation of Learning with Latent Variables	129
7.4.1	Discovering Latent Groups in Social Media	131
7.4.1.1	Latent Group Model	134
7.4.1.2	Predictive Performance	136
7.4.1.3	Discovered Groups	136
7.4.2	Latent User Attributes in Trust Networks	140
7.4.3	Image Completion	143
8	Conclusion	149
A	Proof of Theorem 2	153
B	Additional Results on Learning with Latent Variables	161
	Bibliography	168

Chapter 1: Introduction

In many problems in artificial intelligence and machine learning, the domains are rich and structured, with many interdependent elements that are best modeled jointly. Examples include social networks, biological networks, the Web, natural language, computer vision, sensor networks, and so on. The key to accurately modeling such domains is to capture the dependencies induced by their structures. For example, friends in a social network are more likely to share common interests, pages on the Web that link to each other are more likely to share a common topic, and in computer vision what is happening in one frame of video is often correlated with what is happening in adjacent frames. In applications that require making inferences about such domains, constructing a joint model that incorporates these structural dependencies is generally more accurate than attempting to model elements of the domain independently.

Machine learning subfields such as structured prediction (BakIr et al., 2007), statistical relational learning (Getoor and Taskar, 2007), and inductive logic programming (Muggleton and De Raedt, 1994) all seek to model structural dependencies. These three interrelated disciplines are connected by their focus on high-dimensional models in which the unknowns depend on each other in rich and com-

plex ways. Structured prediction (SP) and statistical relational learning (SRL) are closely related and tend to focus on probabilistic models, enabling them to reason under uncertainty. SRL emphasizes constructing probabilistic models using relational formalisms, such as first-order predicate logic. Inductive logic programming (ILP) also uses first-order logic to define rules for inferring the consequences entailed from a set of propositions. See Chapter 2 for a more detailed discussion on SP, SRL, ILP, and their connections.

This thesis focuses on a fundamental challenge in SP: the need for models that are both expressive and scalable. In the worst case, inference in probabilistic models is NP-hard (Shimony, 1994), but this does not mean that the problem is hopeless. Many restricted classes of models admit tractable inference, and approximation techniques for inference in general models often work well in practice. Currently, practitioners must navigate this landscape of specialized models and approximation techniques to find the tools that will capture their problem domains as well as possible while sacrificing as little scalability as possible. This search is challenging because there is generally an inverse relationship between expressivity and scalability. New tools are needed that achieve a good balance between expressivity and scalability across a wide range of problems.

I address this challenge by first unifying three different approaches for scalable inference with structured predictors based on logical formalisms. The first approach is randomized algorithms for MAX SAT, the classic problem of finding a Boolean assignment to variables that maximizes the weighted sum of satisfied clauses in a model. Goemans and Williamson (1994) introduced a convex inference objective

for finding rounding probabilities for each variable such that a randomized rounding procedure will produce a .75-optimal solution to this NP-hard problem in expectation. The procedure can also be derandomized to find a single, .75-optimal discrete solution. The second approach is local consistency relaxation (Wainwright and Jordan, 2008) for discrete Markov random fields with potentials defined using Boolean logic. Again, a convex inference objective is defined, this time based on variational approximations to maximum a posteriori (MAP) inference, i.e., inferring the most probable assignment to the variables. The resulting, possibly fractional solutions are a relaxed solution to MAP inference in the discrete graphical model. Finally, the third approach is Łukasiewicz logic (Klir and Yuan, 1995), a fuzzy logic developed for reasoning about vague or uncertain concepts, e.g., whether something is “bright” or “warm” or “old.” Łukasiewicz logic interprets logical propositions not as true or false, but as continuously valued in the $[0, 1]$ interval. If the logical clauses in the MAX SAT problem are interpreted using Łukasiewicz logic instead of Boolean logic, it defines a new, convex optimization problem for reasoning.

I show that all three approaches actually optimize the same inference objective, unifying these previously distinct interpretations of convex inference for SP. Unifying them is significant because it means that a single modeling framework and set of algorithms can be used to scalably reason and learn in a broad range of structured domains containing both discrete and continuous information. Developing such a modeling framework and algorithms is the subject of the rest of this thesis.

My next contribution is therefore to take the unified inference objective, generalize it, and use it as the energy function of a new kind of probabilistic graphical

model defined over continuous variables, which I call *hinge-loss Markov random fields* (HL-MRFs) because their MAP inference objective is always a weighted sum of hinge losses. HL-MRFs are log-linear models defined over continuous variables in the $[0, 1]$ interval with hinge functions (a maximum of a linear function and zero) for potentials. They extend the unified inference objective in several ways to make it more versatile. First, the potentials are allowed to be arbitrary hinge functions, not just the hinges induced by the three unified interpretations of logical models. This generalization allows HL-MRFs to capture additional types of structural dependencies. Second, I allow linear constraints to be added to the model, which is crucial for modeling many domain elements such as multinomial variables and functional or partial functional relationships. (This is equivalent to allowing potentials to have infinite weights. Third, HL-MRF potentials can be squared, making them piecewise-quadratic instead of piecewise-linear, which creates smooth tradeoffs between conflicting structural dependencies rather than completely favoring the most influential ones.

An important question at this point is what the semantics of these new continuous variables are. They can be interpreted using any of the three semantics (rounding probabilities, pseudomarginals, or fuzzy truth values) used to derive them. However, the additional modeling features I add to derive HL-MRFs—which I show lead to increased functionality and predictive performance—also separates them from their original interpretations. For example, even if features like squared potentials sometimes perform better in practice, using them means that the rounding guarantees of Goemans and Williamson (1994) no longer apply. I therefore take an

agnostic approach to the semantics of these variables. Their interpretation depends on the particular model and application. For example, in a node labeling problem, it might be most useful to interpret the continuous variables as rounding probabilities to obtain discrete final predictions. In a link prediction task, in which the existing relationships to predict are often sparse, it might be best to interpret the continuous values as confidences and order the predictions from most confident to least confident. The results can then be evaluated using ranking measures like area under the precision-recall curve (AuPR) or area under the receiver operating characteristic curve (AuROC), which better characterize performance in sparse prediction problems. The continuous values can also simply represent naturally continuous quantities, like sensor data or pixel intensities.

With this range of possible interpretations, it is important that HL-MRFs generalize approaches based on logical formalisms because logic is a powerful tool for easily applying SP to a wide range of problems. One way to think of a structured domain is as a set of entities, broadly construed, connected by relationships. Mapping the relationships to predicates and the entities to constants enables users to model these domains with logic. A ground atom (a predicate combined with constants for arguments) can represent a relationship involving one or more entities. An SP framework can then map the ground atoms back to variables representing whether each corresponding relationship holds in the domain. Continuing with this idea, logical operators are a shorthand language for defining structural dependencies among the variables represented by atoms. For example, if one variable is true, i.e., a relationship exists in a structured domain, when another variable is true, this can

be indicated by a logical implication.

This convenient formalism for defining structured predictors becomes even more powerful when extended to use first-order logic, in which logical variables can be used as arguments to predicates in logical clauses, acting as placeholders for sets of constants. Then, structural dependencies can be compactly specified for large sets of relationships. One popular SRL framework that uses this idea is Markov logic networks (Richardson and Domingos, 2006), which maps ground atoms to Boolean random variables and logical clauses to potentials in a discrete Markov random field (MRF). Since HL-MRFs generalize approaches to scalable inference in structured predictors defined using logical clauses, they can also be defined using a logical language.

My third contribution is to bring the modeling power and flexibility of logic-based languages to HL-MRFs by introducing a probabilistic programming language called *probabilistic soft logic* (PSL). PSL is a language for compactly specifying families of HL-MRFs using a syntax based on first-order logic. A PSL program is a template that induces a HL-MRF for a given relational data set based on the general structural dependencies described in the program. PSL enables users to easily construct, test, and refine rich models for SP. Logical clauses are mapped to hinge-loss potentials. Additional syntax for arithmetic relationships allows for the specification of a broader set of structural dependencies, including linear constraints. PSL’s syntax makes it easy to define a number of useful features that commonly occur in structured domains, including constraints on the domain and range of relationships, e.g., functional and partial functional relationships, structural

dependencies based on the similarity of entities, priors on variables for which there is no other evidence, blocks and canopies for improving computational performance, and aggregates, which allow variables to be defined as functions of sets of other variables.

PSL is closely related to Markov logic networks (MLNs), in that they both use syntaxes based on first-order logic to define variants of MRFs. Whereas MLNs define discrete MRFs over Boolean variables, PSL defines HL-MRFs over continuous variables. As I show in this thesis, HL-MRFs are much more scalable than discrete MRFs, so the advantage of PSL is that it gives users the benefits of HL-MRFs for structured prediction. There are also some syntactic differences between PSL and MLNs, which I discuss in Section 4.2.

After introducing HL-MRFs and PSL, I introduce a complete stack of algorithms for performing inference and learning with them. The first is an algorithm for MAP inference, the problem of finding a single most-probable assignment to the variables. This problem is fundamental because it is used to make predictions and as a subroutine of many learning algorithms. Scalability is therefore crucial. By design, MAP inference is always a convex optimization, so in principle any off-the-shelf convex optimization toolkit could be used. However, standard general-purpose methods like interior-point methods do not scale well to large-scale structure prediction (Yanover et al., 2006). I introduce a MAP inference algorithm based on consensus optimization that decomposes the problem into many independent subproblems and iterates to combine their solutions into a global solution. The algorithm uses a technique called the alternating direction method of multipliers (ADMM) (Glowin-

ski and Marrocco, 1975; Gabay and Mercier, 1976; Boyd et al., 2011) to derive message-passing updates that guarantee convergence to the MAP state. It is much faster than using an interior-point method, as I show empirically, because it leverages the sparse dependency structure common in SP. MAP inference for HL-MRFs easily scales to models with hundreds of thousands of variables and over a million dependencies, finding the solution for such models in roughly a minute using commodity hardware. I also show that it enables solving the local consistency relaxation for discrete logic-based models more accurately than the current leading approach, dual decomposition (Sontag et al., 2011).

I then turn to supervised learning, considering how to learn the parameters of HL-MRFs from training data. I introduce three learning algorithms that optimize different learning objectives. The first optimizes the maximum likelihood objective. This objective contains an intractable expectation term, so it can be approximated with the MAP state, which is easily found for HL-MRFs. Doing so can be interpreted as using a generalization of the perceptron algorithm for SP. The second learning algorithm maximizes the pseudolikelihood of the training data (Besag, 1975). Maximum pseudolikelihood estimation has the advantage of not requiring joint inference. It learns by conditioning each variable on the true values of the variables on which it depends and estimating expectations with respect to these one-dimensional distributions, which can be much faster computationally. While sometimes this approach suffers from difficulty learning long-range dependencies—in which important structural dependencies are not included in the model directly, but instead modeled as a chain of dependencies—it nevertheless is an important

option for users to have. The third learning algorithm maximizes a large-margin objective Taskar et al. (2004) that generalizes the large-margin objective for binary classification used to train support vector machines (Vapnik, 2000). Large-margin learning trains the model to separate the probability assigned to the training data from other, nearby points in the prediction space.

With these learning algorithms in hand, I evaluate their performance on a range of SP tasks. The first task is node labeling, the problem of classifying entities when their labels depend on the labels of related entities. As a specific instance of this problem, I consider the problem of classifying documents based on topic in citation networks. I show that HL-MRFs are as accurate as traditional, discrete MRFs but take only a small fraction of the time to make predictions. The second task is trust prediction in social networks, which is an instance of link labeling, where the structure of the network is again known, but the task is now to assign labels to the edges as opposed to nodes. Here the labels correspond to whether a trusts or distrusts the opinions of another user. Again, HL-MRFs are as accurate as discrete MRFs, but much faster. Then I study link prediction, the task of predicting the relationships among entities. An important problem in this category is preference prediction, in which there are two types of entities, users and content, and the relationships represent how strongly each user will enjoy each piece of content. Here the continuous variables of HL-MRFs are valuable because they can directly represent the non-Boolean nature of user preferences. The continuous values in the predictions can be sorted per user to construct ordered lists of content recommendations. I show that HL-MRFs can match the performance of state-of-the-art Bayesian

probabilistic matrix factorization (Salakhutdinov and Mnih, 2008) on this problem, while being very easy to construct and interpret. Finally, I compare HL-MRFs to a range of deep learning methods for image completion, showing that they can match or surpass the performance of a range of methods. Again, the continuous values of HL-MRFs make them very easy to apply to this problem because they can directly represent pixel intensity. Together, this range of experiments shows that HL-MRFs and PSL are general-purpose tools, easily applied to many problems and able to scale much better than existing methods.

So far, these learning algorithms and experiments have assumed fully supervised training, i.e., the training data contains labels for all the variables in the model. Often, however, it is useful or necessary to construct a model that contains latent variables, variables for which there are no training labels available during learning. Latent variables can represent many important types of information in structured domains. One kind is hard-to-acquire information, such as the opinions of all users in a social network (so a model could be learned using just the opinions of users that are available). Another kind is information that is impossible to acquire because it is inherently unobservable, such as whether someone is trustworthy. While there might be observable indications of such a latent trait, it is inherently a subjective judgement. Latent variables can also capture abstract, lower dimensional representations of a problem's feature space. For example, in an image completion task, latent variables can represent a mixture of archetypical images to use to complete the given partial image.

Latent variables are therefore very important tools for SP models, and it is im-

portant that HL-MRFs and PSL support them. Traditional approaches to learning structured predictors with latent variables, such as “hard” expectation maximization or latent structured support vector machines (LSSVM) (Yu and Joachims, 2009), are equally applicable to HL-MRFs as they are to discrete MRFs. However, these approaches are constrained by the fact that they require repeated inferences to update beliefs over the latent variables and compute the gradient of the learning objective. Recently, several methods have been introduced that overcome the challenge of learning when computing the gradient of the objective requires inference. For fully-supervised learning, large-margin methods can use the dual of loss-augmented inference to form a joint convex minimization (Taskar et al., 2005; Meshi et al., 2010). Schwing et al. (2012a) extended this idea to latent-variable learning for discrete MRFs, using a method specifically formulated to pass messages corresponding to the discrete states of the variables.

I use a similar approach to solve the inference bottleneck for HL-MRFs, solving the challenge that arises in continuous models of dealing with intractable expectations and entropy terms in the learning objective. This new algorithm is called paired-dual learning (PDL), and it makes learning with latent variables extremely fast by interleaving ADMM inference updates with updates to the parameters. It is so fast that it often converges to an accurate model before traditional methods have made a single parameter update.

I show that PDL has a big impact on the computational cost of learning with latent variables by evaluating it on a range of models. On these tasks, PDL converges to accurate models in as little as 10% of the time required by traditional

methods, often before the traditional methods can make a single parameter update. The first task I consider is community detection in social media. I show that HL-MRFs can discover interesting associations among supporters of political candidates, language, and media outlets by modeling latent political support in Twitter data. PDL makes training these models much faster. I then return to trust prediction in social networks, showing that modeling latent user attributes like whether each user is trusting or trustworthy can significantly improve performance over the fully supervised model. Again, PDL makes learning this superior model much faster. Finally, I revisit image completion to address the problem of learning to construct more realistic faces by modeling real facial structure. When predicting the bottom half of a face from the top, the fully supervised models evaluated in the previous chapter rely on superficial symmetries that lead to good pixel-level accuracy but produce very unnatural images. Learning latent, archetypical face structures leads to more realistic images, and once again PDL makes doing so much faster. PDL completes the stack of inference and learning algorithms that make HL-MRFs and PSL general-purpose tools for SP.

Together, HL-MRFs and PSL address the challenge of balancing scalability and expressivity by offering users a class of models that admits very fast inference and is easily applicable to a wide range of problems. In summary, this thesis demonstrates that they are as accurate as traditional methods for SP, while being much more scalable. HL-MRFs unite and generalize several fundamental approaches to artificial intelligence and machine learning problems: randomized algorithms for MAX SAT, relaxed inference for probabilistic graphical models, and fuzzy logic.

PSL provides users with a friendly and powerful interface for defining models for many different structured domains. I introduce algorithms for inference, supervised learning, and learning with latent variables; and demonstrate that these algorithms enable HL-MRFs and PSL to model many domains accurately and scalably. Together, this package of new tools enables users to solve machine learning problems with a level of accuracy at a scale that was not previously possible.

Material in this thesis has been published at top-tier venues for research on machine learning and artificial intelligence. The equivalence between MAX SAT and local consistency relaxation was published in *Artificial Intelligence & Statistics* (Bach et al., 2015b). Work on MAP inference was published in *Neural Information Processing Systems* (Bach et al., 2012). Additional work on MAP inference and work on supervised learning appeared in *Uncertainty in Artificial Intelligence* (Bach et al., 2013). Paired-dual learning appeared at the *International Conference on Machine Learning* (Bach et al., 2015a).

Chapter 2: Related Work

Researchers in artificial intelligence and machine learning have long been interested in predicting interdependent unknowns using structural dependencies. Some of the earliest work in this area is inductive logic programming (ILP) (Muggleton and De Raedt, 1994), in which structural dependencies are described with first-order logic. Using first-order logic has several advantages. First, it can capture many types of dependencies among variables, such as correlations, anti-correlations, and implications. Second, it can compactly specify dependencies that hold across many different sets of propositions by using variables as wildcards that match entities in the data. These features enable the construction of intuitive, general-purpose models that are easily applicable or adapted to different domains. Inference for ILP finds the propositions that satisfy a query, consistent with a relational knowledge base. However, ILP is limited by its difficulty in coping with uncertainty. Standard ILP approaches only model dependencies which hold universally, and such dependencies are rare in real-world data.

Another broad area of research, probabilistic methods, directly models uncertainty over unknowns. Probabilistic graphical models (PGMs) (Koller and Friedman, 2009) are a family of formalisms for specifying joint distributions over inter-

dependent unknowns through graphical structures. The graphical structure of a PGM generally represents conditional independence relationships among random variables. Explicitly representing conditional independence relationships allows a distribution to be more compactly parametrized. For example, in the worst case, a discrete distribution could be represented by an exponentially large table over joint assignments to the random variables. However, describing the distribution in smaller, conditionally independent pieces can be much more compact. Similar benefits apply to continuous distributions. Algorithms for probabilistic inference and learning can also operate over the conditionally independent pieces described by the graph structure. They are therefore straightforward to apply to a wide variety of distributions. Categories of PGMs include Markov random fields (MRFs), Bayesian networks (BNs), and dependency networks (DNs). Constructing PGMs often requires careful design, and models are usually constructed for single tasks and data sets.

More recently, researchers have sought to combine the advantages of relational and probabilistic approaches, creating the field of *statistical relational learning* (SRL) (Getoor and Taskar, 2007). SRL techniques build probabilistic models of relational data, i.e., data composed of entities and relationships connecting them. Relational data is obviously most often described using a relational calculus, but SRL techniques are also equally applicable to similar categories of data that go by other names, such as graph data or network data. Modeling relational data is inherently complicated by the large number of interconnected and overlapping structural dependencies that are typically present. This has motivated two directions of

work. The first direction is algorithmic, seeking inference and learning methods that scale up to very high dimensional models. The other direction is both user-oriented and—as a growing body of evidence shows—supported by learning theory, seeking formalisms for compactly specifying entire groups of dependencies in the model that share both form and parameters. Specifying these grouped dependencies, often in the form of templates via a domain-specific language, is convenient for users. Most often in relational data the structural dependencies hold without regard to the identities of entities, instead being induced by an entity’s class (or classes) and the structure of its relationships with other entities. Therefore, many SRL models and languages give users the ability to specify dependencies in this abstract form and ground out models over specific data sets based on these definitions. In addition to convenience, recent work in learning theory says that repeated dependencies with tied parameters can be the key to generalizing from a few—or even one—large, structured training example(s) (London et al., 2013a).

A related field to SRL is *structured prediction* (SP) (BakIr et al., 2007), which generalizes the traditional tasks of binary and multiclass classification using a 0-1 loss to the task of predicting from a structured space. The loss function used during learning and evaluation is generalized to a task-appropriate loss function that scores disagreement between predictions and the true structures. Often, models for structured prediction take the form of energy functions that are linear in their parameters. Therefore, prediction with such models is equivalent to MAP inference for MRFs. A distinct branch of SP is learn-to-search methods, in which the problem is decomposed into a series of one-dimension prediction problems. The challenge

is to learn a good order in which to predict the components of the structure, so that each one-dimension prediction problem can be conditioned on the most useful information. Examples of learn-to-search methods include incremental structured perceptron (Collins and Roark, 2004), SEARN (Daumé III et al., 2009), DAgger (Ross et al., 2011), and AggreVaTe (Ross and Bagnell, 2014).

In this thesis we focus on SP methods that perform joint prediction directly. Better understanding the differences and relative advantages of joint-prediction methods and learn-to-search methods is an important direction for future work. In the rest of the chapter we survey models and domain-specific languages for SP and SRL (Section 2.1), inference methods (Section 2.2), and learning methods (Section 2.3).

2.1 Models and Languages

SP and SRL encompass many approaches. One broad area of work—of which PSL is a part—uses first-order logic and other relational formalisms to specify templates for PGMs. Probabilistic relational models (Friedman et al., 1999) define templates for BNs in terms of a database schema, and they can be grounded out over instances of that schema to create BNs. Relational dependency networks (Neville and Jensen, 2007) template RNs using server query language (SQL) queries over a relational schema. Markov logic networks (MLNs) (Richardson and Domingos, 2006) use first-order logic to define Boolean MRFs. Each logical clause in a first-order knowledge base is a template for a set of potentials when the MLN is grounded out over a set of

propositions. Whether each proposition is true is a Boolean random variable, and the potential has a value of one when the corresponding ground clause is satisfied by the propositions and zero when it is not. (MLNs are formulated such that higher values of the energy function are more probable.) Clauses can either be weighted, in which case the potential has the weight of the clause that templated it, or unweighted, in which case it must hold universally, as in ILP. In these ways, MLNs are similar to PSL. Whereas MLNs are defined over Boolean variables, PSL is a templating language for HL-MRFs, which are defined over continuous variables. However, these continuous variables can be used to model discrete quantities. See Section 3.1 for more information on the relationships between HL-MRFs and discrete MRFs, and Section 6.4 for empirical comparisons between the two. As we show, HL-MRFs and PSL scale much better while retaining the rich expressivity and accuracy of their discrete counterparts. In addition, HL-MRFs and PSL can reason directly about continuous data.

Another logic-based language is probabilistic similarity logic (Broecheler et al., 2010a), which is a predecessor of probabilistic soft logic. Probabilistic similarity logic defines distributions for reasoning about similarity in relational data, many of which can be viewed as instances of HL-MRFs. Probabilistic soft logic is a more general language for defining HL-MRFs, so it can be applied to both discrete data and continuous data, including similarity. Further, probabilistic soft logic provides syntax for defining arithmetic rules (Section 4.1.5) that can express many important modeling dependencies like domain and range rules (Section 4.3.1) and aggregates (Section 4.3.5).

PSL is part of a broad family of probabilistic programming languages (Gordon et al., 2014). The goals of probabilistic programming and SRL often overlap. Probabilistic programming seeks to make constructing probabilistic models easy for the end user, and separate model specification from the development of inference and learning algorithms. If algorithms can be developed for the entire space of models covered by a language, then it is easy for users to experiment with including and excluding different model components. It also makes it easy for existing models to benefit from improved algorithms. Separation of model specification and algorithms is also useful in SRL for the same reasons. In this thesis we emphasize designing algorithms that are flexible enough to support the full class of HL-MRFs. Examples of probabilistic programming languages include IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), ProbLog (De Raedt et al., 2007), Church (Goodman et al., 2008), Figaro (Pfeffer, 2009), and FACTORIE (McCallum et al., 2009).

2.2 Inference

Whether viewed as MAP inference for an MRF or SP without probabilistic semantics, searching over a structured space to find the optimal prediction is an important but difficult task. It is NP-hard in general (Shimony, 1994), so much work has focused on approximations and identifying classes of problems for which it is tractable. A well-studied approximation technique is local consistency relaxation (LCR) (Wainwright and Jordan, 2008). Inference is first viewed as an equivalent optimization over the realizable expected values of the potentials, called the marginal

polytope. When the variables are discrete and each potential is an indicator that a subset of variables is in a certain state, this optimization becomes a linear program. Each variable in the program is the marginal probability that a variable is a particular state or the variables associated with a potential are in a particular joint state. The marginal polytope is then the set of marginal probabilities that are globally consistent. The number of linear constraints required to define the marginal polytope is exponential in the size of the problem, however, so the linear program has to be relaxed in order to be tractable. In a local consistency relaxation, the marginal polytope is relaxed to the local polytope, in which the marginals over variables and potential states are only locally consistent in the sense that each marginal over potential states sums to the marginal distributions over the associated variables.

A large body of work has focused on solving the LCR objective quickly. Typically, off-the-shelf convex optimization methods do not scale well for large graphical models and structured predictors (Yanover et al., 2006), so a large branch of research has investigated highly scalable message-passing algorithms. One approach is dual decomposition (DD) Sontag et al. (2011), which solves a problem dual to the LCR objective. Many DD algorithms use coordinate descent, such as TRW-S (Kolmogorov, 2006), MSD (Werner, 2007), MPLP (Globerson and Jaakkola, 2007), and ADLP (Meshi and Globerson, 2011), Other DD algorithms use subgradient-based approaches (e.g., Jojic et al., 2010; Komodakis et al., 2011; Schwing et al., 2012b).

Another approach to solving the LCR objective uses message-passing algorithms to solve the problem directly in its primal form. One well-known algorithm is that of Ravikumar et al. (2010), which uses proximal optimization, a general

approach that iteratively improves the solution by searching for nearby improvements. The authors also provide rounding guarantees for when the relaxed solution is integral, i.e., the relaxation is tight, allowing the algorithm to converge faster. Another message-passing algorithm that solves the primal objective is AD³ (Martins et al., 2015), which uses the alternating direction method of multipliers (ADMM). AD³ optimizes objective (3.10) for binary, pairwise MRFs and supports the addition of certain deterministic constraints on the variables. A third example of a primal message-passing algorithm is APLP (Meshi and Globerson, 2011), which is the primal analog of ADLP. Like AD³, it uses ADMM to optimize the objective.

Other approaches to approximate inference include tighter linear programming relaxations (Sontag et al., 2008, 2012). These tighter relaxations enforce local consistency on variable subsets that are larger than individual variables, which makes them *higher-order local consistency relaxations*. Mezuman et al. (2013) developed techniques for special cases of higher-order relaxations, such as when the MRF contains cardinality potentials, in which the probability of a configuration depends on the number of variables in a particular state. Some papers have also explored non-linear convex programming relaxations, e.g., Ravikumar and Lafferty (2006) and Kumar et al. (2006).

Previous analyses have identified particular subclasses whose local consistency relaxations are tight, i.e., the maximum of the relaxed program is exactly the maximum of the original problem. These special classes include graphical models with tree-structured dependencies, models with submodular potential functions, models encoding bipartite matching problems, and those with *nand* potentials and

perfect graph structures (Wainwright and Jordan, 2008; Schrijver, 2003; Jebara, 2009; Foulds et al., 2011). Researchers have also studied performance guarantees of other subclasses of the first-order local consistency relaxation. Kleinberg and Tardos (2002) and Chekuri et al. (2005) considered the metric labeling problem. Feldman et al. (2005) used the local consistency relaxation to decode binary linear codes.

In this thesis we examine the classic problem of MAX SAT—finding a joint Boolean assignment to a set of propositions that maximizes the sum of a set of weighted clauses that are satisfied—as an instance of SP. Researchers have also considered approaches to solving MAX SAT other than the one we study, the randomized algorithm of Goemans and Williamson (1994). One line of work focusing on convex programming relaxations has obtained stronger rounding guarantees than Goemans and Williamson (1994) by using nonlinear programming, e.g., Asano and Williamson (2002) and references therein. Other work does not use the probabilistic method but instead searches for discrete solutions directly, e.g., Mills and Tsang (2000), Larrosa et al. (2008), and Choi et al. (2009). We note that one such approach, that of Wah and Shang (1997), is essentially a type of DD formulated for MAX SAT. A more recent approach blends convex programming and discrete search via mixed integer programming (Davies and Bacchus, 2013). Additionally, Huynh and Mooney (2009) introduced a linear programming relaxation for MLNs inspired by MAX SAT relaxations, but the relaxation of general Markov logic provides no known guarantees on the quality of solutions.

2.3 Learning

Taskar et al. (2004) connected SP and PGMs by showing how to train MRFs with large-margin estimation, a generalization of the large-margin objective for binary classification used to train support vector machines (Vapnik, 2000). Large-margin learning is a well-studied approach to train structured predictors because it directly incorporates the structured loss function into a convex upper bound on the true objective: the regularized expected risk. The learning objective is to find the parameters with smallest norm such that a linear combination of feature functions assign a better score to the training data than all other possible predictions. The amount by which the score of the correct prediction must beat the score of other predictions is scaled using the structured loss function. The objective is therefore encoded as a norm minimization problem subject to many linear constraints, one for each possible prediction in the structured space.

Structured SVMs (Tsochantaridis et al., 2005) extend large-margin estimation to a broad class of structured predictors and admit a tractable cutting-plane learning algorithm. This algorithm will terminate in a number of iterations linear in the size of the problem, and so the computational challenge of large-margin learning for structured prediction comes down to the task of finding the most violated constraint in the learning objective. This can be accomplished by optimizing the energy function plus the loss function. In other words, the task is to find the structure that is the best combination of being favored by the energy function but disfavored by the loss function. Often, the loss function decomposes over the components of the

prediction space, so the combined energy function and loss function can often be viewed as simply the energy function of another structured predictor that is equally challenging or easy to optimize, such as when the space of structures is a set of discrete vectors and the loss function is the Hamming distance.

It is common during large-margin estimation that no setting of the parameters can predict all the training data without error. In this case, the training data is said to not be separable, again generalizing the notion of linear separability in the feature space from binary classification. The solution to this problem is to add slack variables to the constraints that require the training data to be assigned the best score. The magnitude of the slack variables are penalized in the learning objective, so estimation must trade off between the norm of the parameters and violating the constraints. Joachims et al. (2009) extend this formulation to a “one slack” formulation, in which a single slack variable is used for all the constraints across all training examples, which is more efficient. We use this framework for large-margin estimation for HL-MRFs in Section 6.3.

The repeated inferences required for large-margin learning, one to find the most-violated constraint at each iteration, can become computationally expensive. Therefore researchers have explored speeding up learning by interleaving the inference problem with the learning problem. In the cutting-plane formulation discussed above, the objective is equivalently a saddle-point problem, with the solution at the minimum with respect to the parameters and the maximum with respect to the inference variables. Taskar et al. (2005) proposed dualizing the inner inference problem to form a joint minimization. For SP problems with a tight duality gap, i.e.,

the dual problem has the same optimal value as the primal problem, this approach leads to an equivalent, convex optimization that can be solved for all variables simultaneously. In other words, the learning and most-violated constraint problems are solved simultaneously, greatly reducing training time. For problems with non-tight duality gaps, e.g., MAP inference in general, discrete MRFs, Meshi et al. (2010) showed that that the same principle can be applied by using approximate inference algorithms like dual decomposition to bound the primal objective.

Chapter 3: Hinge-Loss Markov Random Fields

In this chapter we present hinge-loss Markov random fields (HL-MRFs), a new kind of probabilistic graphical model designed to make structured prediction highly scalable. We first unify several approaches to scalable structured prediction based on logical formalisms in Section 3.1. We show that different approaches to approximating NP-hard inference problems in discrete models via convex inference lead to the same inference objective, and that this objective is also the exact inference objective for continuous structured predictors defined using Łukasiewicz logic, a fuzzy logic for continuous values. To the best of our knowledge, we are the first to show their equivalence. That they are the equivalent is significant because it means that the same class of models and set of algorithms can be used to reason scalably in a wide range of discrete and continuous domains. In Section 3.2 we generalize this unified inference objective to add additional functionality, and in Section 3.3 we use the generalized inference objective as an energy function to define HL-MRFs.

3.1 Unifying Inference in Logic-Based Models

In many structured domains, propositional and first-order logics are useful tools for describing the intricate dependencies that connect the unknown variables. However,

these domains are usually noisy; dependencies among the variables do not always hold. To address this, logical semantics can be incorporated into probability distributions to create models that capture both the structure and the uncertainty in machine learning tasks. One common way to do this is to use logic to define feature functions in a probabilistic model. We focus on Markov random fields (MRFs), a popular class of probabilistic graphical models for rich, structured data. Informally, an MRF is a distribution that assigns probability mass using a scoring function that is a weighted combination of feature functions called potentials. We will use logical clauses to define these potentials. We first define MRFs more formally to introduce necessary notation:

Definition 1 *Let $\mathbf{x} = (x_1, \dots, x_n)$ be a vector of random variables and let $\boldsymbol{\phi} = (\phi_1, \dots, \phi_m)$ be a vector of potentials, where each potential $\phi_j(\mathbf{x})$ assigns configurations of the variables a real-valued score. Also, let $\mathbf{w} = (w_1, \dots, w_n)$ be a vector of real-valued weights. Then, a **Markov random field** is a probability distribution of the form*

$$P(\mathbf{x}) \propto \exp(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x})) . \quad (3.1)$$

In an MRF, the potentials should capture how the domain behaves, assigning higher scores to more probable configurations of the variables. If a modeler does not know how the domain behaves, the potentials should capture how it might behave, so that a learning algorithm can find weights that lead to accurate predictions. Logic provides an excellent formalism for defining such potentials in structured and relational domains.

We now introduce some notation to make this logic-based approach more formal. Consider a set of logical clauses $\mathbf{C} = \{C_1, \dots, C_m\}$ where each clause $C_j \in \mathbf{C}$ is a disjunction of literals and each literal is a variable x or its negation $\neg x$ drawn from the variables \mathbf{x} such that each variable $x_i \in \mathbf{x}$ appears at most once in C_j . Let I_j^+ (resp. I_j^-) $\subset \{1, \dots, n\}$ be the set of indices of the variables that are not negated (resp. negated) in C_j . Then C_j can be written as

$$\left(\bigvee_{i \in I_j^+} x_i \right) \vee \left(\bigvee_{i \in I_j^-} \neg x_i \right). \quad (3.2)$$

Logical clauses of this form are very expressive because they can be viewed equivalently as implications from conditions to consequences:

$$\bigwedge_{i \in I_j^-} x_i \implies \bigvee_{i \in I_j^+} x_i. \quad (3.3)$$

This “if-then” reasoning is intuitive and can describe many dependencies in structured data. Further, multiple clauses can together express dependencies that cannot be expressed in a single clause, such as multiple sets of conditions implying one set of possible consequences, or one set of conditions implying multiple sets of possible consequences. See Section 4.2 for more information on the expressivity of models defined with disjunctive clauses.

Assuming we have a logical knowledge base \mathbf{C} describing a structured domain, we can embed it in an MRF by defining each potential ϕ_j using a corresponding clause C_j . If an assignment to the variables \mathbf{x} satisfies C_j , then we let $\phi_j(\mathbf{x})$ equal 1,

and we let it equal 0 otherwise. For our subsequent analysis we now let $w_j \geq 0$ ($\forall j = 1, \dots, m$). The resulting MRF preserves the structured dependencies described in \mathbf{C} , but enables much more flexible modeling. Clauses no longer have to hold always, and the model can express our uncertainty over different possible worlds. The weights express how strongly we expect each corresponding clause to hold; the higher the weight, the more probable that it is true according to the model.

This notion of embedding weighted, logical knowledge bases in MRFs is an appealing one. For example, Markov logic networks (Richardson and Domingos, 2006) are a popular formalism that induce MRFs from weighted first-order knowledge bases.¹ Given a data set, the first-order clauses are grounded using the constants in the data to create the set of propositional clauses \mathbf{C} . Each propositional clause has the weight of the first-order clause from which it was grounded. In this way, a weighted, first-order knowledge base can compactly specify an entire family of MRFs for a structured machine-learning task.

Although we now have a method for easily defining rich, structured models for a wide range of problems, there is a new challenge: finding a most probable assignment to the variables, i.e., MAP inference, is NP-hard (Shimony, 1994; Garey et al., 1976). This means that (unless P=NP) our only hope for performing tractable inference is to perform it approximately. Observe that MAP inference for an MRF

¹Markov logic networks also allow clauses that contain conjunctions, as well as negative or infinite weights, and therefore subsume the models discussed in this section. However, the full class of Markov logic networks does not admit any known polynomial-time approximation schemes for MAP inference.

defined by \mathcal{C} is the integer linear program

$$\begin{aligned} \arg \max_{\mathbf{x} \in \{0,1\}^n} P(\mathbf{x}) &\equiv \arg \max_{\mathbf{x} \in \{0,1\}^n} \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) \\ &\equiv \arg \max_{\mathbf{x} \in \{0,1\}^n} \sum_{C_j \in \mathcal{C}} w_j \min \left\{ \sum_{i \in I_j^+} x_i + \sum_{i \in I_j^-} (1 - x_i), 1 \right\}. \end{aligned} \quad (3.4)$$

While this program is intractable, it does admit convex programming relaxations.

In this section, we show how convex programming can be used to perform tractable inference in MRFs defined by weighted knowledge bases. We first discuss in Section 3.1.1 an approach developed by Goemans and Williamson (1994) that views MAP as an instance the classic MAX SAT problem and relaxes it to a convex program from that perspective. This approach has the advantage of providing strong guarantees on the quality of the discrete solutions it obtains. However, it has the disadvantage that general-purpose convex programming toolkits do not scale well to relaxed MAP inference for large graphical models (Yanover et al., 2006). In Section 3.1.2 we then discuss a seemingly distinct approach, local consistency relaxation, with complementary advantages and disadvantages: it offers highly scalable message-passing algorithms but come with no quality guarantees. We then unite these approaches by proving that they solve equivalent optimization problems with identical solutions. Then, in Section 3.1.3, we show that the unified inference objective is also equivalent to exact MAP inference if the knowledge base \mathcal{C} is interpreted not with Boolean logic but with Łukasiewicz logic, an infinite-valued logic for reasoning about naturally continuous quantities such as similarity, vague or fuzzy

concepts, and real-valued data.

That these three interpretations all lead to the same inference objective—whether reasoning about discrete or continuous information—is extremely useful. To the best of our knowledge, we are the first to show their equivalence. It indicates that the same modeling formalism, inference algorithms, and learning algorithms can be used to reason scalably and accurately about both discrete and continuous information in structured domains. We will generalize the unified inference objective in Section 3.2 to derive hinge-loss MRFs, and in the rest of the thesis we will develop a probabilistic programming language and algorithms that realize the goal of a scalable and accurate framework for structured data, both discrete and continuous.

3.1.1 MAX SAT Relaxation

One approach to approximating objective (3.4) is to use relaxation techniques developed in the randomized algorithms community for the MAX SAT problem. Formally, the MAX SAT problem is to find a Boolean assignment to a set of variables that maximizes the total weight of satisfied clauses in a knowledge base composed of disjunctive clauses annotated with nonnegative weights. In other words, objective (3.4) is an instance of MAX SAT. Randomized approximation algorithms can be constructed for MAX SAT by independently rounding each Boolean variable x_i to true with probability p_i . Then, the expected weighted satisfaction \hat{w}_j of a clause C_j is

$$\hat{w}_j = w_j \left(1 - \prod_{i \in I_j^+} (1 - p_i) \prod_{i \in I_j^-} p_i \right), \quad (3.5)$$

also known as a (weighted) noisy-or function, and the expected total score \hat{W} is

$$\hat{W} = \sum_{C_j \in \mathcal{C}} w_j \left(1 - \prod_{i \in I_j^+} (1 - p_i) \prod_{i \in I_j^-} p_i \right). \quad (3.6)$$

Optimizing \hat{W} with respect to the rounding probabilities would give the exact MAX SAT solution, so this randomized approach hasn't made the problem any easier yet, but Goemans and Williamson (1994) showed how to bound \hat{W} below with a tractable linear program.

To approximately optimize \hat{W} , associate with each Boolean variable x_i a corresponding continuous variable \hat{y}_i with domain $[0, 1]$. Then let $\hat{\mathbf{y}}^*$ be the optimum to the linear program

$$\arg \max_{\hat{\mathbf{y}} \in [0, 1]^n} \sum_{C_j \in \mathcal{C}} w_j \min \left\{ \sum_{i \in I_j^+} \hat{y}_i + \sum_{i \in I_j^-} (1 - \hat{y}_i), 1 \right\}. \quad (3.7)$$

Observe that objectives (3.4) and (3.7) are of the same form, except that the variables are relaxed to the unit hypercube in objective (3.7). Goemans and Williamson (1994) showed that if p_i is set to \hat{y}_i^* for all i , then $\hat{W} \geq .632 Z^*$, where Z^* is the optimal total weight for the MAX SAT problem. If each p_i is set using any function in a special class, then this lower bound improves to a .75 approximation. One simple example of such a function is

$$p_i = \frac{1}{2} \hat{y}_i^* + \frac{1}{4}. \quad (3.8)$$

In this way, objective (3.7) leads to an *expected* .75 approximation of the MAX SAT solution.

The method of conditional probabilities (Alon and Spencer, 2008) can find a single Boolean assignment that achieves at least the expected score from a set of rounding probabilities, and therefore at least .75 of the MAX SAT solution when objective (3.7) and function (3.8) are used to obtain them. Each variable x_i is greedily set to the value that maximizes the expected weight over the unassigned variables, conditioned on either possible value of x_i and the previously assigned variables. This greedy maximization can be applied quickly because, in many models, variables only participate in a small fraction of the clauses, making the change in expectation quick to compute for each variable. Specifically, referring to the definition of \hat{W} (3.6), the assignment to x_i only needs to maximize over the clauses C_j in which x_i participates, i.e., $i \in I_j^+ \cup I_j^-$, which is usually a small set.

This approximation is powerful because it is a tractable linear program that comes with strong guarantees on solution quality. However, even though it is tractable, general-purpose convex optimization toolkits do not scale well to large MAP problems. In the following subsection, we unify this approximation with a complementary one developed in the probabilistic graphical models community.

3.1.2 Local Consistency Relaxation

Another approach to approximating objective (3.4) is to apply a relaxation developed for Markov random fields called local consistency relaxation (Wainwright and

Jordan, 2008). This approach starts by viewing MAP inference as an equivalent optimization over marginal probabilities.² For each $\phi_j \in \phi$, let θ_j be a marginal distribution over joint assignments \mathbf{x}_j . For example, $\theta_j(\mathbf{x}_j)$ is the probability that the subset of variables associated with potential ϕ_j is in a particular joint state \mathbf{x}_j . Also, let $x_j(i)$ denote the setting of the variable with index i in the state \mathbf{x}_j .

With this variational formulation, inference can be relaxed to an optimization over the *first-order local polytope* \mathbb{L} . Let $\boldsymbol{\mu} = (\mu_1, \dots, \mu_n)$ be a vector of probability distributions, where $\mu_i(k)$ is the marginal probability that x_i is in state k . The first-order local polytope is

$$\mathbb{L} \triangleq \left\{ (\boldsymbol{\theta}, \boldsymbol{\mu}) \geq \mathbf{0} \left| \begin{array}{l} \sum_{\mathbf{x}_j | x_j(i)=k} \theta_j(\mathbf{x}_j) = \mu_i(k) \quad \forall i, j, k \\ \sum_{\mathbf{x}_j} \theta_j(\mathbf{x}_j) = 1 \quad \forall j \\ \sum_{k=0}^{K_i-1} \mu_i(k) = 1 \quad \forall i \end{array} \right. \right\}, \quad (3.9)$$

which constrains each marginal distribution θ_j over joint states \mathbf{x}_j to be consistent only with the marginal distributions $\boldsymbol{\mu}$ over individual variables that participate in the potential ϕ_j .

MAP inference can then be approximated with the *first-order local consistency relaxation*:

$$\arg \max_{(\boldsymbol{\theta}, \boldsymbol{\mu}) \in \mathbb{L}} \sum_{j=1}^m w_j \sum_{\mathbf{x}_j} \theta_j(\mathbf{x}_j) \phi_j(\mathbf{x}_j), \quad (3.10)$$

which is an upper bound on the true MAP objective. Much work has focused on

²This treatment is for discrete MRFs. We have omitted a discussion of continuous MRFs for conciseness.

solving the first-order local consistency relaxation for large-scale MRFs, which we discuss further in Section 2. These algorithms are appealing because they are well-suited to the sparse dependency structures common in MRFs, so they can scale to very large problems. However, in general, the solutions are fractional, and there are no guarantees on the approximation quality of a tractable discretization of these fractional solutions.

We show that for MRFs with potentials defined by \mathbf{C} and nonnegative weights, local consistency relaxation is equivalent to MAX SAT relaxation.

Theorem 2 *For an MRF with potentials corresponding to disjunctive logical clauses and associated nonnegative weights, the first-order local consistency relaxation of MAP inference is equivalent to the MAX SAT relaxation of Goemans and Williamson (1994). Specifically, any partial optimum $\boldsymbol{\mu}^*$ of objective (3.10) is an optimum $\hat{\boldsymbol{y}}^*$ of objective (3.7), and vice versa.*

We prove Theorem 2 in Appendix A. Our proof analyzes the local consistency relaxation to derive an equivalent, more compact optimization over only the variable pseudomarginals $\boldsymbol{\mu}$ that is identical to the MAX SAT relaxation. Theorem 2 is significant because it shows that the rounding guarantees of MAX SAT relaxation also apply to local consistency relaxation, and the scalable message-passing algorithms developed for local consistency relaxation also apply to MAX SAT relaxation.

3.1.3 Łukasiewicz Logic

The previous two subsections showed that the same convex program can approximate MAP inference in discrete, logic-based models, whether viewed from the perspective of MAX SAT or of probabilistic models. In this subsection, we show that this convex program can also be used to reason about naturally continuous information, such as similarity, vague or fuzzy concepts, and real-valued data. Instead of interpreting the clauses \mathcal{C} using Boolean logic, we can interpret them using Łukasiewicz logic (Klir and Yuan, 1995), which extends Boolean logic to infinite-valued logic in which the propositions \mathbf{x} can take truth values in the continuous interval $[0, 1]$. Extending truth values to a continuous domain enables them to represent concepts that are vague, in the sense that they are often neither completely true nor completely false. For example, the propositions that a sensor value is high, two entities are similar, or a protein is highly expressed can all be captured in a more nuanced manner in Łukasiewicz logic. We can also use the now continuous valued \mathbf{x} to represent quantities that are naturally continuous (scaled to $[0, 1]$), such as actual sensor values, similarity scores, and protein expression levels. The ability to reason about continuous values is very valuable, as many important applications are not entirely discrete.

The extension to continuous values requires a corresponding extended interpretation of the logical operators \wedge (conjunction), \vee (disjunction), and \neg (negation). The Łukasiewicz t-norm and t-co-norm are \wedge and \vee operators that correspond to

the Boolean logic operators for integer inputs (along with the negation operator \neg):

$$x_1 \wedge x_2 = \max \{x_1 + x_2 - 1, 0\} \quad (3.11)$$

$$x_1 \vee x_2 = \min \{x_1 + x_2, 1\} \quad (3.12)$$

$$\neg x = 1 - x . \quad (3.13)$$

The analogous MAX SAT problem for Łukasiewicz logic is therefore

$$\arg \max_{\mathbf{x} \in [0,1]^n} \sum_{C_j \in \mathcal{C}} w_j \min \left\{ \sum_{i \in I_j^+} x_i + \sum_{i \in I_j^-} (1 - x_i), 1 \right\} , \quad (3.14)$$

which is identical in form to objective (3.7). Therefore, if an MRF is defined over continuous variables with domain $[0, 1]^n$ and the logical knowledge base \mathcal{C} defining the potentials is interpreted using Łukasiewicz logic, then *exact* MAP inference is identical to finding the optimum using the unified, relaxed inference objective derived for Boolean logic in the previous two subsections. This shows the equivalence of all three approaches: MAX SAT relaxation, local consistency relaxation, and Łukasiewicz logic.

3.2 Generalizing Convex Inference

We have shown that a single convex program can be used to reason scalably and accurately about both discrete and continuous information. In this subsection, we generalize this inference objective to derive *hinge-loss Markov random fields* (HL-MRFs), a new kind of probabilistic graphical model. HL-MRFs will preserve convex,

scalable MAP inference and the expressivity of logic-based modeling, but will additionally support an even richer space of dependencies. To begin, we will define HL-MRFs as density functions over continuous variables $\mathbf{y} = (y_1, \dots, y_n)$ with joint domain $[0, 1]^n$, but we will remain agnostic about the semantics of these variables. Since we are generalizing the interpretations explored in Section 3.1, their MAP states can be viewed as rounding probabilities or pseudomarginals, or they can represent naturally continuous information. More generally, they can be viewed simply as degrees of belief, confidences, or rankings of possible states; and they can describe discrete, continuous, or mixed domains.

To derive HL-MRFs, we will generalize the unified inference objective of Section 3.1 in several ways, which we restate for our semantics-agnostic variables:

$$\arg \max_{\mathbf{y} \in [0, 1]^n} \sum_{C_j \in \mathbf{C}} w_j \min \left\{ \sum_{i \in I_j^+} y_i + \sum_{i \in I_j^-} (1 - y_i), 1 \right\}. \quad (3.15)$$

For now, we are still assuming that the objective terms are defined using a weighted knowledge base \mathbf{C} , but we will quickly drop this requirement. To do so, we examine one term in isolation. Observe that the maximum value of any unweighted term is 1, which is achieved when a linear function of the variables is at least 1. We say that the term is *satisfied* whenever this occurs. When a term is unsatisfied, we can refer to its *distance to satisfaction*, how far it is from achieving its maximum value. Also observe that we can rewrite the optimization explicitly in terms of distances

to satisfaction:

$$\arg \min_{\mathbf{y} \in [0,1]^n} \sum_{C_j \in \mathcal{C}} w_j \max \left\{ 1 - \sum_{i \in I_j^+} y_i - \sum_{i \in I_j^-} (1 - y_i), 0 \right\}, \quad (3.16)$$

so that the objective is equivalently to minimize the total weighted distance to satisfaction. Each unweighted objective term now measures how far the linear constraint

$$1 - \sum_{i \in I_j^+} y_i - \sum_{i \in I_j^-} (1 - y_i) \leq 0 \quad (3.17)$$

is from being satisfied.

3.2.1 Relaxed Linear Constraints

With this view of each term as a relaxed linear constraint, we can easily generalize them to arbitrary linear constraints. We no longer require that the inference objective be defined using only logical clauses, and instead each term can be defined using any function $\ell_j(\mathbf{y})$ that is linear in \mathbf{y} . Then, the new inference objective is

$$\arg \min_{\mathbf{y} \in [0,1]^n} \sum_{j=1}^m w_j \max \{ \ell_j(\mathbf{y}), 0 \}. \quad (3.18)$$

Now each term represents the distance to satisfaction of a linear constraint $\ell_j(\mathbf{y}) \leq 0$. That constraint could be defined using logical clauses as discussed above, or it could be defined using other knowledge about the domain. The weight w_j indicates how important it is to satisfy a constraint relative to others by scaling the distance to

satisfaction. The higher the weight, the more distance to satisfaction is penalized. Additionally, two relaxed inequality constraints, $\ell_j(\mathbf{y}) \leq 0$ and $-\ell_j(\mathbf{y}) \leq 0$, can be combined to represent a relaxed equality constraint $\ell_j(\mathbf{y}) = 0$.

3.2.2 Hard Linear Constraints

Now that our inference objective admits arbitrary relaxed linear constraints, it is natural to also allow hard constraints that must be satisfied at all times. Hard constraints are important modeling tools. They enable groups of variables to represent a multinomial or categorical variable, mutually exclusive possibilities, and functional or partial functional relationships. Hard constraints can also represent background knowledge about the domain, restricting the domain to regions that are feasible in the real world. Additionally, they can encode more complex components such as defining a random variable as an aggregate over other unobserved variables, which we discuss further in Section 4.3.5.

We can think of including hard constraints as allowing a weight w_j to take an infinite value. Again, two inequality constraints can be combined to represent an equality constraint. However, when we introduce an inference algorithm for HL-MRFs in Section 5, it will be useful to treat hard constraints separately from relaxed ones, and further, treat hard inequality constraints separately from hard equality constraints. Therefore, in the definition of HL-MRFs, we will define these three components separately.

3.2.3 Generalized Hinge-Loss Functions

The objective terms measuring each constraint’s distance to satisfaction are hinge losses. There is a flat region, on which the distance to satisfaction is 0, and an angled region, on which the distance to satisfaction grows linearly away from the hyperplane $\ell_j(\mathbf{y}) = 0$. This loss function is very useful—as we discuss in the previous section, it is a bound on the expected loss in the discrete setting, among other things—but it is not appropriate for all modeling situations.

A piecewise-linear loss function makes MAP inference “winner take all,” in the sense that it is preferable to fully satisfy the most highly weighted objective terms completely before reducing the distance to satisfaction of terms with lower weights. For example, consider the following optimization problem:

$$\arg \min_{y_1 \in [0,1]} 5 \max \{y_1, 0\} + 2 \max \{1 - y_1, 0\} . \quad (3.19)$$

The optimizer is $y_1 = 0$ because the term with weight 5 that prefers $y_1 = 0$ overrules the term with weight 2 that prefers $y_1 = 1$. The result does not indicate any ambiguity or uncertainty, but if the two objective terms are potentials in a probabilistic model, it is sometimes preferable that the result reflect this conflicting information. We can change the inference problem so that it smoothly trades off satisfying conflicting objective terms by squaring the hinge losses. Observe that in the modified problem

$$\arg \min_{y_1 \in [0,1]} 5 (\max \{y_1, 0\})^2 + 2 (\max \{1 - y_1, 0\})^2 \quad (3.20)$$

the optimizer is $y_1 = \frac{2}{7}$, reflecting the relative influence of the two loss functions.

Another advantage of squared hinge-loss functions is that they can behave more intuitively in the presence of hard constraints. Consider the problem

$$\arg \min_{(y_1, y_2) \in [0, 1]^2} \max \{0.9 - y_1, 0\} + \max \{0.6 - y_2, 0\} \tag{3.21}$$

such that $y_1 + y_2 \leq 1$.

The first term prefers $y_1 \geq 0.9$, the second term prefers $y_2 \geq 0.6$, and the constraint requires that y_1 and y_2 are mutually exclusive. Such problems are very common and arise when conflicting evidence of different strengths support two mutually exclusive possibilities. The evidence values 0.9 and 0.6 could come from many sources, including base models trained to make independent predictions on individual random variables, domain-specialized similarity functions, or sensor readings. For this problem, any solution $y_1 \in [0.4, 0.9]$ and $y_2 = 1 - y_1$ is an optimizer. This includes counterintuitive optimizers like $y_1 = 0.4$ and $y_2 = 0.6$, even though the evidence supporting y_1 is stronger. Again, squared hinge losses ensure the optimizers better reflect the relative strength of evidence. For the problem

$$\arg \min_{(y_1, y_2) \in [0, 1]^2} (\max \{0.9 - y_1, 0\})^2 + (\max \{0.6 - y_2, 0\})^2 \tag{3.22}$$

such that $y_1 + y_2 \leq 1$,

the only optimizer is $y_1 = 0.65$ and $y_2 = 0.35$, which is a more informative solution.

We therefore complete our generalized inference objective by allowing either hinge-loss or squared hinge-loss functions. Users of HL-MRFs have the choice of either one for each potential, depending on which is appropriate for their task.

3.3 Definition

We can now formally state the full definition of HL-MRFs. They are defined so that a MAP state is a solution to the generalized inference objective derived in the previous subsection. We state the definition in a conditional form for later convenience, but this definition is fully general since the vector of conditioning variables may be empty.

Definition 3 *Let $\mathbf{y} = (y_1, \dots, y_n)$ be a vector of n variables and $\mathbf{x} = (x_1, \dots, x_{n'})$ a vector of n' variables with joint domain $\mathbf{D} = [0, 1]^{n+n'}$. Let $\boldsymbol{\phi} = (\phi_1, \dots, \phi_m)$ be a vector of m continuous potentials of the form*

$$\phi_j(\mathbf{y}, \mathbf{x}) = (\max\{\ell_j(\mathbf{y}, \mathbf{x}), 0\})^{p_j} \quad (3.23)$$

where ℓ_j is a linear function of \mathbf{y} and \mathbf{x} and $p_j \in \{1, 2\}$. Let $\mathbf{c} = (c_1, \dots, c_r)$ be a vector of r linear constraint functions associated with index sets denoting equality constraints \mathcal{E} and inequality constraints \mathcal{I} , which define the feasible set

$$\tilde{\mathbf{D}} = \left\{ (\mathbf{y}, \mathbf{x}) \in \mathbf{D} \left| \begin{array}{l} c_k(\mathbf{y}, \mathbf{x}) = 0, \forall k \in \mathcal{E} \\ c_k(\mathbf{y}, \mathbf{x}) \leq 0, \forall k \in \mathcal{I} \end{array} \right. \right\}. \quad (3.24)$$

For $(\mathbf{y}, \mathbf{x}) \in \mathbf{D}$, given a vector of m nonnegative free parameters, i.e., weights, $\mathbf{w} = (w_1, \dots, w_m)$, a **constrained hinge-loss energy function** $f_{\mathbf{w}}$ is defined as

$$f_{\mathbf{w}}(\mathbf{y}, \mathbf{x}) = \sum_{j=1}^m w_j \phi_j(\mathbf{y}, \mathbf{x}) . \quad (3.25)$$

We now define HL-MRFs by placing a probability density over the inputs to a constrained hinge-loss energy function. Note that we negate the hinge-loss energy function so that states with lower energy are more probable, in contrast with Definition 1. This change is made for later notational convenience.

Definition 4 A **hinge-loss Markov random field** P over random variables \mathbf{y} and conditioned on random variables \mathbf{x} is a probability density defined as follows: if $(\mathbf{y}, \mathbf{x}) \notin \tilde{\mathbf{D}}$, then $P(\mathbf{y}|\mathbf{x}) = 0$; if $(\mathbf{y}, \mathbf{x}) \in \tilde{\mathbf{D}}$, then

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{w}, \mathbf{x})} \exp(-f_{\mathbf{w}}(\mathbf{y}, \mathbf{x})) \quad (3.26)$$

where

$$Z(\mathbf{w}, \mathbf{x}) = \int_{\mathbf{y} | (\mathbf{y}, \mathbf{x}) \in \tilde{\mathbf{D}}} \exp(-f_{\mathbf{w}}(\mathbf{y}, \mathbf{x})) \, d\mathbf{y} . \quad (3.27)$$

In the rest of this thesis, we will explore how to use HL-MRFs to solve a wide range of structured machine learning problems. We first introduce a probabilistic programming language that makes HL-MRFs easy to define for large, rich domains.

Chapter 4: Probabilistic Soft Logic

In this chapter we introduce a general-purpose probabilistic programming language, *probabilistic soft logic* (PSL). PSL allows HL-MRFs to be easily applied to a broad range of structured machine learning problems by defining *templates* for potentials and constraints. In models for structured data, there are very often repeated patterns of probabilistic dependencies. A few of the many examples include the strength of ties between similar people in social networks, the preference for triadic closure when predicting transitive relationships, and the “exactly one active” constraints on functional relationships. Often, to make graphical models that are both easy to define and which generalize across different data sets, these repeated dependencies are defined using templates. Each template defines an abstract dependency, such as the form of a potential function or constraint, along with any necessary parameters, such as the weight of the potential, each of which has a single value across all dependencies defined by that template. Given input data, an undirected graphical model is constructed from a set of templates by first identifying the random variables in the data and then “grounding out” each template by introducing a potential or constraint into the graphical model for each subset of random variables to which the template applies.

A PSL program is written in a first-order syntax and defines a class of HL-MRFs that are parameterized by the input data. PSL provides a natural interface to represent hinge-loss potential templates using two types of rules: logical rules and arithmetic rules. Logical rules are based on the mapping from logical clauses to hinge-loss potentials introduced in Section 3.1. Arithmetic rules provide additional syntax for defining an even wider range of hinge-loss potentials and hard constraints.

4.1 Definition

In this section we define PSL. Our definition covers the essential functionality that should be supported by all implementations, but many extensions are possible. The PSL syntax we describe can capture a very wide range of HL-MRFs, but new settings and scenarios could motivate the development of additional syntax to make the construction of different kinds of HL-MRFs more convenient.

4.1.1 Preliminaries

We begin with a high-level definition of PSL programs.

Definition 5 *A PSL program is a set of rules, each of which is a template for hinge-loss potentials or hard linear constraints. When grounded over a base of ground atoms, a PSL program induces a HL-MRF conditioned on any specified observations.*

In the PSL syntax, many of components are named using *identifiers*, which are strings that begin with a letter (from the set $\{\mathbf{A}, \dots, \mathbf{Z}, \mathbf{a}, \dots, \mathbf{z}\}$), followed by zero

or more letters, numeric digits, or underscores.

PSL programs are grounded out over data, so the universe over which to ground must be defined.

Definition 6 *A constant is a string that denotes an element in the universe over which a PSL program is grounded.*

Constants are the elements in a universe of discourse. They can be entities or attributes. For example, the constant "person1" can denote a person, the constant "Adam" can denote a person's name, and the constant "30" can denote a person's age. In PSL programs, constants are written as strings in double or single quotes. Constants use backslashes as escape characters, so they can be used to encode quotes within constants. It is assumed that constants are unambiguous, i.e., different constants refer to different entities and attributes.¹ Groups of constants can be represented using variables.

Definition 7 *A variable is an identifier for which constants can be substituted.*

Variables and constants are the arguments to logical predicates. Together, they are generically referred to as terms.

Definition 8 *A term is either a constant or a variable.*

Terms are connected by relationships called predicates.

Definition 9 *A predicate is a relation defined by a unique identifier and a positive*

¹Note that ambiguous references to underlying entities can be modeled by using different constants for different references and representing whether they refer to the same underlying entity as a predicate.

integer called its arity, which denotes the number of terms it accepts as arguments.

Every predicate in a PSL program must have a unique identifier as its name.

We refer to a predicate using its identifier and arity appended with a slash. For example, the predicate `Friends/2` is a binary predicate, i.e., taking two arguments, which represents whether two constants are friends. As another example, the predicate `Name/2` can relate a person to the string that is that person's name. As a third example, the predicate `EnrolledInClass/3` can relate two entities, a student and professor, with an additional attribute, the subject of the class.

Predicates and terms are combined to create atoms.

Definition 10 *An atom is a predicate combined with a sequence of terms of length equal to the predicate's arity. This sequence is called the atom's arguments. An atom with only constants for arguments is called a ground atom.*

Ground atoms are the basic units of reasoning in PSL. Each represents an unknown or observation of interest and can take any value in $[0, 1]$. For example, the ground atom `Friends("person1", "person2")` represents whether "person1" and "person2" are friends. Atoms that are not ground are placeholders for sets of ground atoms. For example, the atom `Friends(X, Y)` stands for all ground atoms that can be obtained by substituting constants of type `Person` for variables `X` and `Y`.

4.1.2 Inputs

As we have already stated, PSL defines templates for hinge-loss potentials and hard linear constraints that are grounded out over a data set to induce a HL-MRF. We now describe how that data set is represented and provided as the inputs to a PSL program. The first two inputs are two sets of predicates, a set \mathbb{C} of *closed* predicates, the atoms of which are completely observed, and a set \mathbb{O} of *open* predicates, the atoms of which may be unobserved. The third input is the *base* \mathcal{A} , which is the set of all ground atoms under consideration. All atoms in \mathcal{A} must have a predicate in either \mathbb{C} or \mathbb{O} . These are the atoms which can be substituted into the rules and constraints of a PSL program, and each will later be associated with a HL-MRF random variable with domain $[0, 1]$. The final input is a function $\mathcal{O} : \mathcal{A} \rightarrow [0, 1] \cup \{\emptyset\}$ that maps the ground atoms in the base to either an observed value in $[0, 1]$ or a symbol \emptyset indicating that it is unobserved. The function \mathcal{O} is only valid if all atoms with a predicate in \mathbb{C} are mapped to a $[0, 1]$ value. Note that this makes the sets \mathbb{C} and \mathbb{O} redundant in a sense, since they can be derived from \mathcal{A} and \mathcal{O} , but it will be convenient later to have \mathbb{C} and \mathbb{O} explicitly defined.

Ultimately, the method for specifying PSL’s inputs is implementation specific, since different choices make it more or less convenient for different scenarios. In this thesis, we will assume that \mathbb{C} , \mathbb{O} , \mathcal{A} , and \mathcal{O} exist and remain agnostic about how they were specified. However, to make this aspect of using PSL more concrete, we will describe one possible method for defining them here.

Our example method for specifying PSL’s inputs is text-based. The first sec-

tion of the text input is a definition of the constants in the universe, which are grouped into types. An example universe definition is given:

```
Person = {"Alexis", "Bob", "Claudia", "Don"}
```

```
Professor = {"Alexis", "Bob"}
```

```
Student = {"Claudia", "Don"}
```

```
Subject = {"Computer_Science", "Statistics"}
```

This universe includes six constants, four with two types ("Alexis", "Bob", "Claudia", and "Don") and two with one type ("Computer_Science" and "Statistics").

The next section of input is the definition of predicates. Each predicate includes the types of constants it takes as arguments and whether it is closed. For example, we can define predicates for an advisor-student relationship prediction task as follows:

```
Advises(Professor, Student)
```

```
Department(Person, Subject) (closed)
```

```
EnrolledInClass(Student, Subject, Professor) (closed)
```

In this case, there is one open predicate (`Advises`) and two closed predicates (`Department` and `EnrolledInClass`).

The final section of input is any associated observations. They can be specified

in a list, for example:

```
Advices("Alexis", "Don") = 1  
  
Department("Alexis", "Computer_Science") = 1  
  
Department("Bob", "Computer_Science") = 1  
  
Department("Claudia", "Statistics") = 1  
  
Department("Don", "Statistics") = 1
```

In addition, values for atoms with the `EnrolledInClass` predicate could also be specified. If a ground atom does not have a specified value, it will have a default observed value of 0 if its predicate is closed or remain unobserved if its predicate is open.

We now describe how this text input is processed into the formal inputs \mathbb{C} , \mathbb{O} , \mathcal{A} , and \mathcal{O} . First, each predicate is added to either \mathbb{C} or \mathbb{O} based on whether it is annotated with the `(closed)` tag. Then, for each predicate in \mathbb{C} or \mathbb{O} , ground atoms of that predicate are added to \mathcal{A} with each sequence of constants as arguments that can be created by selecting a constant of each of the predicate's argument types. For example, assume that the input file contains a single predicate definition

```
Category(Document, Cat_Name)
```

where the universe is `Document = {"d1", "d2"}` and `Cat_Name = {"politics", "sports"}`.

Then,

$$\mathcal{A} = \left\{ \begin{array}{l} \text{Category("d1", "politics"),} \\ \text{Category("d1", "sports"),} \\ \text{Category("d2", "politics"),} \\ \text{Category("d2", "sports")} \end{array} \right\}. \quad (4.1)$$

Finally, we define the function \mathcal{O} . Any atom in the explicit list of observations is mapped to the given value. Then, any remaining atoms in \mathcal{A} with a predicate in \mathbb{C} are mapped to 0 and any with a predicate in \mathbb{O} are mapped to \emptyset .

Before moving on, we also note that PSL implementations can support predicates and atoms that are defined functionally. Such predicates can be thought of as a type of closed predicates. Their observed values are defined as a function of their arguments. One of the most common examples is inequality, atoms of which can be represented with the shorthand infix operator \neq . For example, the following atom has a value of 1 when two variables A and B are replaced with different constants and 0 when replaced with the same:

$$A \neq B$$

Such functionally defined predicates can be implemented without requiring their values over all arguments to be specified by the user.

4.1.3 Rules and Grounding

Before introducing the syntax and semantics of specific PSL rules, we define the grounding procedure that induces HL-MRFs in general. Given the inputs \mathbb{C} , \mathbb{O} , \mathcal{A} , and \mathcal{O} , PSL induces a HL-MRF $P(\mathbf{y}|\mathbf{x})$ as follows. First, each ground atom $a \in \mathcal{A}$ is associated with a random variable with domain $[0, 1]$. If $\mathcal{O}(a) = \emptyset$, then the variable is included in the free variables \mathbf{y} , and otherwise it is included in the observations \mathbf{x} with a value of $\mathcal{O}(a)$.

With the variables in the distribution defined, each rule in the PSL program is applied to the inputs and produces hinge-loss potentials or hard linear constraints, which are added to the HL-MRF. In the rest of this subsection, we describe two kinds of PSL rules: logical rules and arithmetic rules.

4.1.4 Logical Rules

The first kind of PSL rule is a logical rule, which is made up of literals.

Definition 11 *A literal is an atom or a negated atom.*

In PSL, the prefix operator $!$ or \sim is used for negation. A negated atom refers to one minus the value of that atom. For example, if `Friends("person1", "person2")` has a value of 0.7, then `!Friends("person1", "person2")` has a value of 0.3.

Definition 12 *A logical rule is a disjunctive clause of literals. Logical rules are either weighted or unweighted. If a logical rule is weighted, it is annotated with a nonnegative weight and optionally a power of two.*

Logical rules express logical dependencies in the model. As in Boolean logic, the negation, disjunction (written as `||` or `|`), and conjunction (written as `&&` or `&`) operators obey De Morgan's Laws. Also, an implication (written as `->` or `<-`) can be rewritten as the negation of the body disjoined with the head. For example

$$\begin{aligned}
 & P1(A, B) \ \&\& \ P2(A, B) \ -> \ P3(A, B) \ || \ P4(A, B) \\
 \equiv & \ !(P1(A, B) \ \&\& \ P2(A, B)) \ || \ P3(A, B) \ || \ P4(A, B) \\
 \equiv & \ !P1(A, B) \ || \ !P2(A, B) \ || \ P3(A, B) \ || \ P4(A, B)
 \end{aligned}$$

Therefore, any formula written as an implication with a literal or conjunction of literals in the body, and a literal or disjunction of literals in the head is also a valid logical rule, because it is equivalent to a disjunctive clause.

There are two kinds of logical rules, weighted or unweighted. A weighted logical rule is a template for a hinge-loss potential that penalizes how far the rule is from being satisfied. A weighted logical rule begins with a nonnegative weight and optionally ends with an exponent of two ($\wedge 2$). For example, the weighted logical rule

```
10 : Advisor(Prof, S) && Department(Prof, Sub) -> Department(S, Sub)
```

has a weight of 10 and induces potentials propagating department membership from advisors to advisees. An unweighted logical rule is a template for a hard linear constraint that requires that the rule always be satisfied. For example, the unweighted

logical rule

$$\text{Friends}(X, Y) \ \&\& \ \text{Friends}(Y, Z) \ \rightarrow \ \text{Friends}(X, Z) \ .$$

induces hard linear constraints enforcing the transitivity of the `Friends/2` predicate. Note the period (`.`) that is used to emphasize that this rule is always enforced and disambiguate it from weighted rules.

A logical rule is grounded out by performing all possible substitutions of ground atoms in the base \mathcal{A} for atoms in the rule, such that the replaced constants agree with the substituted atoms and variables are consistently mapped to the same constants within each grounding. This produces a set of *ground rules*, which are rules containing only ground atoms. Each ground rule will then be interpreted as either a potential or hard constraint in the induced HL-MRF. For notational convenience, we will assume without loss of generality that all the random variables are unobserved, i.e., $\mathcal{O}(a) = \emptyset, \forall a \in \mathcal{A}$. If the input data contain any observations, the following description still applies, except that some free variables will be replaced with observations from \mathbf{x} . The first step in interpreting a ground rule is to map its disjunctive clause to a linear constraint. This is done using the mapping to the unified inference objective derived in Section 3.1. Any ground PSL rule is a disjunction of literals, some of which are negated. Let I^+ be a set of the indices of the variables that correspond to atoms that are not negated in the ground rule, expressed as a disjunctive clause, and, likewise, let I^- be the indices of the variables corresponding

to atoms that are negated. Then, the clause is mapped to the inequality

$$1 - \sum_{i \in I^+} y_i - \sum_{i \in I^-} (1 - y_i) \leq 0. \quad (4.2)$$

If the logical rule that templated the ground rule is weighted with a weight of w and is *not* annotated with $\wedge 2$, then the potential

$$\phi(\mathbf{y}, \mathbf{x}) = \max \left\{ 1 - \sum_{i \in I^+} y_i - \sum_{i \in I^-} (1 - y_i), 0 \right\} \quad (4.3)$$

is added to the HL-MRF with a parameter of w . If the rule is weighted with a weight w and annotated with $\wedge 2$, then the potential

$$\phi(\mathbf{y}, \mathbf{x}) = \left(\max \left\{ 1 - \sum_{i \in I^+} y_i - \sum_{i \in I^-} (1 - y_i), 0 \right\} \right)^2 \quad (4.4)$$

is added to the HL-MRF with a parameter of w . If the rule is unweighted, then the function

$$c(\mathbf{y}, \mathbf{x}) = 1 - \sum_{i \in I^+} y_i - \sum_{i \in I^-} (1 - y_i) \quad (4.5)$$

is added to the set of constraint functions and its index is included in the set \mathcal{I} to define a hard inequality constraint $c(\mathbf{y}, \mathbf{x}) \leq 0$.

As an example of the grounding process, consider the following logical rule:

3 : Friends(A, B) && Friends(B, C) -> Friends(C, A) $\wedge 2$

Imagine that the input data are $\mathbb{C} = \{\}$, $\mathbb{O} = \{\text{Friends}/2\}$,

$$\mathcal{A} = \left\{ \begin{array}{l} \text{Friends}(\text{"p1"}, \text{"p2"}), \\ \text{Friends}(\text{"p1"}, \text{"p3"}), \\ \text{Friends}(\text{"p2"}, \text{"p1"}), \\ \text{Friends}(\text{"p2"}, \text{"p3"}), \\ \text{Friends}(\text{"p3"}, \text{"p1"}), \\ \text{Friends}(\text{"p3"}, \text{"p2"}) \end{array} \right\}, \quad (4.6)$$

and $\mathcal{O}(a) = \emptyset, \forall a \in \mathcal{A}$. Then, the rule will induce six ground rules. One such ground rule is

$$\begin{aligned} 3 : & \text{Friends}(\text{"p1"}, \text{"p2"}) \ \&\& \ \text{Friends}(\text{"p2"}, \text{"p3"}) \\ & \rightarrow \text{Friends}(\text{"p3"}, \text{"p1"}) \ \wedge^2 \end{aligned}$$

which is equivalent to

$$\begin{aligned} 3 : & \ !\text{Friends}(\text{"p1"}, \text{"p2"}) \ \|\ \ !\text{Friends}(\text{"p2"}, \text{"p3"}) \\ & \ \|\ \ \text{Friends}(\text{"p3"}, \text{"p1"}) \ \wedge^2 \end{aligned}$$

If the atoms $\text{Friends}(\text{"p1"}, \text{"p2"})$, $\text{Friends}(\text{"p2"}, \text{"p3"})$, and $\text{Friends}(\text{"p3"}, \text{"p1"})$ correspond to the random variables y_1 , y_2 , and y_3 , respectively, then this

ground rule is interpreted as the weighted hinge-loss potential

$$3 (\max\{y_1 + y_2 - y_3 - 1, 0\})^2 . \quad (4.7)$$

Since the grounding process uses the mapping from Section 3.1, logical rules can be used to reason accurately and efficiently about both discrete and continuous information. They are a convenient method for constructing HL-MRFs with the unified inference objective for weighted logical knowledge bases as their MAP inference objective. They also allow the user to seamlessly incorporate some of the additional features of HL-MRFs, such as squared potentials and hard constraints. Next, we introduce an even more flexible class of PSL rules.

4.1.5 Arithmetic Rules

Arithmetic rules in PSL are more general templates for hinge-loss potentials and hard linear constraints. Like logical rules, they come in weighted and unweighted variants, but instead of using logical operators they use arithmetic operators. In general, an arithmetic rule relates two linear combinations of atoms with a nonstrict inequality or an equality. A simple example enforces the mutual exclusivity of liberal and conservative ideologies:

$$\text{Liberal}(P) + \text{Conservative}(P) = 1 .$$

Just as logical rules are grounded out by performing all possible substitutions of ground atoms, arithmetic rules are grounded out to define potentials and hard constraints over ground atoms. In this example, each possible substitution for `Liberal(P)` and `Conservative(P)` is constrained to sum to 1. Since this is an unweighted arithmetic rule, it defines a hard constraint $c(\mathbf{y}, \mathbf{x})$ and its index will be included in \mathcal{E} because it is an equality constraint.

To make arithmetic rules more flexible and easy to use, we define some additional syntax. The first is a generalized definition of atoms that can be substituted with sums of ground atoms, rather than just a single atom.

Definition 13 *A sum-augmented atom is an atom that takes terms and/or sum variables as arguments. A sum-augmented atom represents the sum of all ground atoms that can be obtained by substituting constants for the sum variables.*

A sum variable is represented by prepending a plus symbol (+) to a variable. For example, the sum-augmented atom

$$\text{Friends}(P, +F)$$

is a placeholder for the sum of all ground atoms in \mathcal{A} that have a given first argument. Sum-augmented atoms are useful because they can describe dependencies without needing to specify the number of atoms that can participate. For example, the arithmetic rule

$$\text{Label}(X, +L) = 1 .$$

says that labels for each constant substituted for X should sum to one, without needing to specify how many possible labels there are.

The substitutions for sum variables can be restricted using select statements.

Definition 14 *A select statement is a logical clause defined for a sum variable in an arithmetic rule. The logical clause contains only atoms with predicates that appear in \mathbb{C} and that take constants, variables that appear in the arithmetic rule, and the sum variable for which it is defined as arguments.*

Select statements restrict the substitutions for a sum variable in the corresponding arithmetic rule by only including substitutions for which the statement evaluates to true. Select statements only affect variables in the first arithmetic rule preceding it, not variables in any other rules. The clauses in select statements are evaluated using Boolean logic. For each ground atom a , it is treated as having a value of 0 if and only if $\mathcal{O}(a) = 0$. Otherwise, it is treated as having a value of 1. For example, imagine that we want to restrict the summation in the following arithmetic rule to only constants that satisfy a property `Property/1`.

$$\text{Link}(X, +Y) \leq 1 .$$

Then, we can add the following select statement:

$$\{Y: \text{Property}(Y)\}$$

Then, the hard linear constraints templated by the arithmetic rule will only sum

over constants substituted for Y such that $\text{Property}(Y)$ is non-zero.

In arithmetic rules, atoms can also be modified with coefficients. These coefficients can be hard-coded, e.g.,

$$0.5 P1(X) + 0.5 P2(X) \geq 1 .$$

or they can use PSL's additional coefficient-defining syntax. The first piece of coefficient syntax is a cardinality function that counts the number of terms substituted for a sum variable. Cardinality is denoted by enclosing a sum variable without the $+$ in pipes. For example, the following arithmetic rule says that the average value of a set of atoms must be at least 0.5:

$$P(+X) / |X| \geq 0.5 .$$

Cardinality functions enable rules that depend on the number of substitutions in order to be scaled correctly, such as averaging.

The second piece of coefficient syntax is built-in coefficient functions. The exact set of supported functions is implementation specific, but standard functions like maximum and minimum should be included. Coefficient functions are prepended with $@$ and use square brackets instead of parentheses to distinguish them from predicates. Coefficient functions can take either scalars or cardinality functions as arguments. For example, the following rule includes a coefficient that is the

maximum of the number of summands and a scalar:

$$\text{Max}[2, |X|] P(+X) \leq 5 .$$

In this example, the coefficient for the $P/1$ atoms will be the maximum of 2 and the number of atoms.

So far we have focused on using arithmetic rules to define templates for hard linear constraints, but they can also be used to define hinge-loss potentials. For example, the arithmetic rule

$$5 : 2 P(+X) \leq 1 \wedge 2$$

is a template for weighted hinge-loss potentials of the form

$$5 \left(\max \left\{ 2 \sum_i y_i - 1, 0 \right\} \right)^2 . \tag{4.8}$$

Note that the weight of 5 is distinct from the coefficients in the linear constraint $\ell(\mathbf{y}, \mathbf{x}) \leq 0$ defining the hinge-loss potential. If the arithmetic rule were an equality instead of an inequality, each grounding would be two hinge-loss potentials, one using $\ell(\mathbf{y}, \mathbf{x}) \leq 0$ and one using $-\ell(\mathbf{y}, \mathbf{x}) \leq 0$. In this way, arithmetic rules can define general hinge-loss potentials.

For completeness, we state the full, formal definition of an arithmetic rule and define its grounding procedure.

Definition 15 *An arithmetic rule is a nonstrict inequality or equality relating two linear combinations of sum-augmented atoms. An arithmetic rule can be annotated with a select statement for each sum variable that restricts its groundings. Arithmetic rules are either weighted or unweighted. If an arithmetic rule is weighted, it is annotated with a nonnegative weight and optionally a power of two.*

An arithmetic rule is grounded out by performing all possible substitutions of ground atoms in the base \mathcal{A} for atoms in the rule, such that the replaced constants agree with the substituted atoms and variables are consistently mapped to the same constants. In addition, sum-augmented atoms are replaced by the appropriate summations over ground atoms (possibly restricted by a corresponding select statement) and the coefficient is distributed across the summands. This leads to a set of ground rules for each arithmetic rule given a set of inputs. If the arithmetic rule is an unweighted inequality, each ground rule can be algebraically manipulated to be of the form $c(\mathbf{y}, \mathbf{x}) \leq 0$. Then $c(\mathbf{y}, \mathbf{x})$ is added to the set of constraint functions and its index is added to \mathcal{I} . If instead the arithmetic rule is an unweighted equality, each ground rule is manipulated to $c(\mathbf{y}, \mathbf{x}) = 0$, $c(\mathbf{y}, \mathbf{x})$ is added to the set of constraint functions, and its index is added to \mathcal{E} . If the arithmetic rule is a weighted inequality with weight w , each ground rule is manipulated to $\ell(\mathbf{y}, \mathbf{x}) \leq 0$ and included as a potential of the form

$$\phi(\mathbf{y}, \mathbf{x}) = \max \{ \ell(\mathbf{y}, \mathbf{x}), 0 \} \tag{4.9}$$

with a weight of w . If the arithmetic rule is a weighted equality with weight w , each

ground rule is again manipulated to $\ell(\mathbf{y}, \mathbf{x}) \leq 0$ and two potentials are included,

$$\phi_1(\mathbf{y}, \mathbf{x}) = \max \{\ell(\mathbf{y}, \mathbf{x}), 0\}, \quad \phi_2(\mathbf{y}, \mathbf{x}) = \max \{-\ell(\mathbf{y}, \mathbf{x}), 0\}, \quad (4.10)$$

each with a weight of w . In either case, if the weighted arithmetic rule is annotated with \wedge^2 , then the induced potentials are squared.

4.2 Expressivity

An important question is the expressivity of PSL, which uses disjunctive clauses with positive weights for its logical rules. Other logic-based languages support different types of clauses, such as Markov logic networks (Richardson and Domingos, 2006), which support clauses with conjunctions and clauses with negative weights. As we discuss in this section, PSL’s logical rules capture a general class of structural dependencies, capable of modeling arbitrary probabilistic relationships among Boolean variables, such as those defined by Markov logic networks. The advantage of PSL is that it defines HL-MRFs, which are much more scalable than discrete MRFs and often just as accurate, as we show in Section 6.4.

The expressivity of PSL is tied to the expressivity of the MAX SAT problem, since they both use the same class of weighted clauses. There are two conditions on the clauses: (1) they have nonnegative weights, and (2) they are disjunctive. We first consider the nonnegativity requirement and show that can actually be viewed as a restriction on the structure of a clause. To illustrate, consider a weighted disjunctive

clause of the form

$$-w \quad : \quad \left(\bigvee_{i \in I_j^+} x_i \right) \vee \left(\bigvee_{i \in I_j^-} \neg x_i \right) . \quad (4.11)$$

If it were part of a generalized MAX SAT problem, in which there were no restrictions on weight sign or clause structure, but the goal were still to maximize the sum of the weights of the satisfied clauses, then this clause could be replaced with an equivalent one without changing the optimizer:

$$w \quad : \quad \left(\bigwedge_{i \in I_j^+} \neg x_i \right) \wedge \left(\bigwedge_{i \in I_j^-} x_i \right) . \quad (4.12)$$

Note that the clause has been changed in three ways: (1) the sign of the weight has been changed, (2) the disjunctions have been replaced with conjunctions, and (3) the literals have all been negated. Due to this equivalence, the restriction on the sign of the weights is subsumed by the restriction on the structure of the clauses. In other words, any set of clauses can be converted to a set with nonnegative weights that has the same optimizer, but it might require including conjunctions in the clauses. It is also easy to verify that if Equation (4.11) is used to define a potential in a discrete MRF, replacing it with a potential defined by (4.12) leaves the distribution unchanged, due to the normalizing partition function.

We now consider the requirement that clauses be disjunctive and illustrate how conjunctive clauses can be replaced by an equivalent set of disjunctive clauses. The idea is to construct a set of disjunctive clauses such that all assignments to the variables are mapped to the same score, plus or minus a constant factor. The

simplest example is replacing a conjunction of two variables

$$w_1 \quad : \quad x_1 \wedge x_2 \tag{4.13}$$

with three disjunctions

$$w_2 \quad : \quad x_1 \vee x_2 \tag{4.14}$$

$$w_2 \quad : \quad \neg x_1 \vee x_2 \tag{4.15}$$

$$w_2 \quad : \quad x_1 \vee \neg x_2 \tag{4.16}$$

where w_2 is chosen to ensure that the optimizer remains the same. This can be done by choosing a constant such that all the weights of each disjunctive clause are equal to the weight of the corresponding conjunctive clause plus that constant. We describe this further in the next paragraph.

These examples can be generalized to a procedure for encoding any Boolean MRF into a set of disjunctive clauses with nonnegative weights. Park (2002) showed that the MAP problem for any discrete Bayesian network can be represented as an instance of MAX SAT. For distributions of bounded factor size, the MAX SAT problem has size polynomial in the number of variables and factors of the distribution. We describe how any Boolean MRF can be represented with disjunctive clauses and nonnegative weights. Given a Boolean MRF with arbitrary potentials defined by mappings from joint states of subsets of the variables to scores, a new MRF is created as follows. For each potential in the original MRF, a new set of

potentials defined by disjunctive clauses is created. A conjunctive clause is created corresponding to each entry in the potential’s mapping with a weight equal to the score assigned by the weighted potential in the original MRF. Then, these clauses are converted to equivalent disjunctive clauses as in the example of Equations (4.11) and (4.12) by also flipping the sign of their weights and negating the literals. Once this is done for all entries of all potentials, what remains is an MRF defined by disjunctive clauses, some of which might have negative weights. We make all weights positive by adding a sufficiently large constant to all weights of all clauses, which leaves the distribution unchanged due to the normalizing partition function.

It is important to note two caveats when converting arbitrary Boolean MRFs to MRFs defined by disjunctive clauses with nonnegative weights. First, the number of clauses required to represent a potential in the original MRF is exponential in the size of the potential. In practice, this is rarely a significant limitation, since MRFs often contain low-degree potentials. The other important point is that the step of adding a constant to all the weights increases the total score of the MAP state. Since the bound of Goemans and Williamson (1994) is relative to this score, the bound is loosened for the original problem the larger the constant added to the weights is. This is to be expected, since even approximating MAP is NP-hard in general (Abdelbar and Hedetniemi, 1998).

We have described how general structural dependencies can be modeled with the logical rules of PSL. It is possible to represent arbitrary logical relationships with them. The process for converting general rules to PSL’s logical rules can be done automatically and made transparent to the user. We have elected to define PSL’s

logical rules without making this conversion automatic to make clear the underlying formalism.

4.3 Modeling Patterns

PSL is a very flexible language, and there are some patterns of usage that come up in many applications. We illustrate some of them in this subsection with a number of examples.

4.3.1 Domain and Range Rules

In many problems, the number of relations that can be predicted among some constants is known. For binary predicates, this background knowledge can be viewed as constraints on the domain (first argument) or range (second argument) of the predicate. For example, it might be background knowledge that each entity, such as a document, has exactly one label. An arithmetic rule to express this is

$$\text{Label}(\text{Document}, +\text{LabelName}) = 1 .$$

The predicate `Label` is said to be *functional*.

Alternatively, sometimes it is the first argument that should be summed over. For example, imagine the task of predicting relationships among students and professors. Perhaps it is known that each student has exactly one advisor. This constraint

can be written as

$$\text{Advisor}(+\text{Professor}, \text{Student}) = 1 .$$

The predicate **Advisor** is said to be *inverse functional*.

Finally, imagine a scenario in which two social networks are being aligned. The goal is to predict whether each pair of people, one from each network, is the same person, which is represented with atoms of the **Same** predicate. Each person aligns with at most one person in the other network, but might not align with anyone. This can be expressed with two arithmetic rules:

$$\text{Same}(\text{Person1}, +\text{Person2}) \leq 1 .$$

$$\text{Same}(+\text{Person1}, \text{Person2}) \leq 1 .$$

The predicate **Same** is said to be both *partial functional* and *partial inverse functional*.

Many variations on these examples are possible. For example, they can be generalized to predicates with more than two arguments. Additional arguments can either be fixed or summed over in each rule. As another example, domain and range rules can incorporate multiple predicates, so that an entity can participate in a fixed number of relations counted among multiple predicates.

4.3.2 Similarity

Many problems require explicitly reasoning about similarity, rather than simply whether entities are the same or different. For example, reasoning with similarity has been explored using kernel methods, such as kFoil (Landwehr et al., 2010) that bases similarity computation on the relational structure of the data. The continuous variables of HL-MRFs make modeling similarity straightforward, and PSL’s support for function predicates make it even easier. For example, in an entity resolution task, the degree to which two entities are believed to be the same might depend on how similar their names are. A rule expressing this dependency is

```
1.0 : Name(P1, N1) && Name(P2, N2) && Similar(N1, N2)
      -> Same(P1, P2)
```

This rule uses the `Similar` predicate to measure similarity. Since it is a function predicate, it can be implemented as one of many different, possibly domain specialized, string similarity functions. Any similarity function that can output values in the range $[0, 1]$ can be used.

4.3.3 Priors

If no potentials are defined over a particular atom, then it is equally probable that it has any value between zero and one. Often, however, it should be most probable that an atom has a value of zero, unless there is evidence that it has a nonzero

value. Since atoms typically represent the existence of some entity, attribute, or relation, this bias promotes sparsity among the things inferred to exist. Further, if there is a potential that prefers that an atom should have a value that is at least some other continuous value, such as when reasoning with similarities as discussed in Section 4.3.2, it should also be more probable that an atom is no higher in value than is necessary to satisfy that potential. To accomplish both these goals, simple “priors” can be used to state that atoms should have low values in the absence of evidence to overrules those priors. A prior in PSL can be a rule consisting of just a negative literal with a small weight. For example, in a link prediction task, imagine that this preference should apply to atoms of the `Link` predicate. A prior is then

$$0.1 : \text{!Link}(A, B)$$

This rule acts as a regularizer on `Link` atoms.

4.3.4 Blocks and Canopies

In many tasks, the number of unknowns can quickly grow large, even for modest amounts of data. For example, in a link prediction task, the goal is to predict relations among entities. The number of possible links grows quadratically with the number of entities. If handled naively, this could make scaling to large data sets difficult, but this problem is often handled by constructing *blocks* (e.g., Newcombe and Kennedy, 1962) or *canopies* (McCallum et al., 2000) over the entities, so that a limited subset of all possible links are actually considered. Blocking partitions the

entities so that only links among entities in the same partition element, i.e., block, are considered. Alternatively, for a finer grained pruning, a canopy is defined for each entity, which is the set of other entities to which it could possibly link. Blocks and canopies can be computed using specialized, domain-specific functions, and PSL can incorporate them by including them as atoms in the bodies of rules. Since blocks can be seen as a special case of canopies, we let the atom `InCanopy(A, B)` be 1 if B is in the canopy or block of A, and 0 if it is not. Including `InCanopy(A, B)` atoms as additional conditions in the bodies of logical rules will ensure that the dependencies only exist between the desired entities.

4.3.5 Aggregates

One of the most powerful features of PSL is its ability to easily define *aggregates*, which are rules that define random variables to be deterministic functions of sets of other random variables. The advantage of aggregates is that they can be used to define dependencies that do not scale in magnitude with the number of groundings in the data. For example, consider a model for predicting interests in a social network. A fragment of a PSL program for doing this is

```
1.0 : Interest(P1, I) && Friends(P1, P2) -> Interest(P2, I)

1.0 : Age(P, "20-29") && Lives(P, "California")

      -> Interest(P, "Surfing")
```

These two rules express the belief that interests are correlated along friendship links in the social network, and also that certain demographic information is predictive of specific interests. The question any domain expert or learning algorithm faces is how strongly each rule should be weighted relative to each other. The challenge of answering this question when using templates is that the number of groundings of the first rule varies from person to person based on the number of friends, while the groundings of the second remain constant (one per person). This variable scaling of the two types of dependencies makes it difficult to find weights that accurately reflect the relative influence each type of dependency should have across people with different numbers of friends.

Using an aggregate can solve this problem of variable scaling. Instead of using a separate ground rule to relate the interest of each friend, we can define that is only grounded once for each person, relating an average interest across all friends to each person's own interests. A PSL fragment for this is

```

1.0 : AverageFriendInterest(P, I) -> Interest(P, I)

AverageFriendInterest(P, I) = Interest(+F, I) / |F| .

{F: Friends(P, F)}

/* Demographic dependencies are also included. */

```

Here the predicate `AverageFriendInterest/2` is an aggregate that is constrained

to be the average amount of interest each friend of a person P has in an interest I . The weight w_1 can now be scaled more accurately relative to other types of features because there is only one grounding per person.

For a more complex example, consider the problem of determining whether two references in the data refer to the same underlying person. One useful feature to use is whether they have similar sets of friends in the social network. Again, a rule could be defined that is grounded out for each friendship pair, but this would suffer from the same scaling issues as the previous example. Instead, we can use an aggregate to directly express how similar the two references' sets of friends are. A function that measures the similarity of two sets A and B is *Jaccard similarity*:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

Jaccard similarity is a nonlinear function, meaning that it cannot be used directly without breaking the log-concavity of HL-MRFs, but we can approximate it with a linear function. We define `SameFriends/2` as an aggregate that approximates

Jaccard similarity (where SamePerson/2 is functional and inverse functional):

$$\text{SameFriends}(A, B) = \text{SamePerson}(+FA, +FB) / \text{@Max}[|FA|, |FB|] .$$

$$\{FA : \text{Friends}(A, FA)\}$$

$$\{FB : \text{Friends}(B, FB)\}$$

$$\text{SamePerson}(+P1, P2) = 1 .$$

$$\text{SamePerson}(P1, +P2) = 1 .$$

The aggregate SameFriends/2 uses the sum of the SamePerson/2 atoms as the intersection of the two sets, and the maximum of the sizes of the two sets of friends as a lower bound on the size of their union.

Chapter 5: MAP Inference

Having defined HL-MRFs and a language for creating them, PSL, we turn to algorithms for inference and learning. The first task we consider is maximum a posteriori (MAP) inference, the problem of finding a most probable assignment to the free variables \mathbf{y} given observations \mathbf{x} . In HL-MRFs, the normalizing function $Z(\mathbf{w}, \mathbf{x})$ is constant over \mathbf{y} and the exponential is maximized by minimizing its negated argument, so the MAP problem is

$$\begin{aligned} \arg \max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) &\equiv \arg \min_{\mathbf{y}} f_{\mathbf{w}}(\mathbf{y}, \mathbf{x}) \\ &\equiv \arg \min_{\mathbf{y} \in [0,1]^n} \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{y}, \mathbf{x}) \end{aligned} \tag{5.1}$$

$$\text{such that } c_k(\mathbf{y}, \mathbf{x}) = 0, \quad \forall k \in \mathcal{E}$$

$$c_k(\mathbf{y}, \mathbf{x}) \leq 0, \quad \forall k \in \mathcal{I} .$$

MAP is a fundamental problem because (1) it is the method we will use to make predictions, and (2) weight learning often requires performing MAP inference many times with different weights (as we discuss in Section 6). Here, HL-MRFs have a distinct advantage over general discrete models, since minimizing $f_{\mathbf{w}}$ is a convex

optimization rather than a combinatorial one. There are many off-the-shelf solutions for convex optimization, the most popular of which are interior-point methods, which have worst-case polynomial time complexity in the number of variables, potentials, and constraints (Nesterov and Nemirovskii, 1994). Although in practice they perform better than their worst-case bounds (Wright, 2005), they do not scale well to big structured prediction problems (Yanover et al., 2006). We therefore introduce a new algorithm for exact MAP inference designed to scale to very large HL-MRFs by leveraging the sparse connectivity structure of the potentials and hard constraints that are typical of models of real-world domains.

5.1 Consensus Optimization Formulation

Our algorithm uses *consensus optimization*, a technique that divides an optimization problem into independent subproblems and then iterates to reach a consensus on the optimum Boyd et al. (2011). Given a HL-MRF $P(\mathbf{y}|\mathbf{x})$, we first construct an equivalent MAP problem in which each potential and hard constraint is a function of different variables. The variables are then constrained to make the new and original MAP problems equivalent. We let $\mathbf{y}_{(L,j)}$ be a copy of the variables in \mathbf{y} that are used in the potential function ϕ_j , $j = 1, \dots, m$ and $\mathbf{y}_{(L,k+m)}$ be a copy of those used in the constraint function c_k , $k = 1, \dots, r$. We refer to the concatenation of all of these vectors as \mathbf{y}_L . We also introduce an indicator function I_k for each constraint function where $I_k \left[c_k(\mathbf{y}_{(L,k+m)}, \mathbf{x}) \right] = 0$ if the constraint is satisfied and infinity if it is not. Likewise, let $I_{[0,1]}$ be an indicator function that is 0 if the input

is in the interval $[0, 1]$ and infinity if it is not. We drop the constraints on the domain of \mathbf{y} , letting them range in principle over \mathbb{R}^n and instead use these indicator functions to enforce the domain constraints. This will make computation easier when the problem is later decomposed. Finally, let $\mathbf{y}_{(C,\hat{i})}$ be the variables in \mathbf{y} that correspond to $\mathbf{y}_{(L,\hat{i})}$, $\hat{i} = 1, \dots, m+r$. Operators between $\mathbf{y}_{(L,\hat{i})}$ and $\mathbf{y}_{(C,\hat{i})}$ are defined element-wise, pairing the corresponding copied variables. Consensus optimization solves the reformulated MAP problem

$$\arg \min_{(\mathbf{y}_L, \mathbf{y})} \sum_{j=1}^m w_j \phi_j(\mathbf{y}_{(L,j)}, \mathbf{x}) + \sum_{k=1}^r I_k [c_k(\mathbf{y}_{(L,k+m)}, \mathbf{x})] + \sum_{i=1}^n I_{[0,1]} [y_i] \quad (5.2)$$

such that $\mathbf{y}_{(L,\hat{i})} = \mathbf{y}_{(C,\hat{i})} \quad \forall \hat{i} = 1, \dots, m+r$.

Inspection shows that problems (5.1) and (5.2) are equivalent.

This reformulation enables us to relax the equality constraints $\mathbf{y}_{(L,\hat{i})} = \mathbf{y}_{(C,\hat{i})}$ in order to divide problem (5.2) into independent subproblems that are easier to solve, using the alternating direction method of multipliers (ADMM) (Glowinski and Marrocco, 1975; Gabay and Mercier, 1976; Boyd et al., 2011). The first step is to form the *augmented Lagrangian* function for the problem. Let $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_{m+r})$ be a concatenation of vectors of Lagrange multipliers. Then the augmented Lagrangian

is

$$\begin{aligned} \mathcal{L}(\mathbf{y}_L, \boldsymbol{\alpha}, \mathbf{y}) &= \sum_{j=1}^m w_j \phi_j(\mathbf{y}_{(L,j)}, \mathbf{x}) + \sum_{k=1}^r I_k[c_k(\mathbf{y}_{(L,k+m)}, \mathbf{x})] + \sum_{i=1}^n I_{[0,1]}[y_i] \\ &\quad + \sum_{\hat{i}=1}^{m+r} \boldsymbol{\alpha}_{\hat{i}}^\top (\mathbf{y}_{(L,\hat{i})} - \mathbf{y}_{(C,\hat{i})}) + \frac{\rho}{2} \sum_{\hat{i}=1}^{m+r} \left\| \mathbf{y}_{(L,\hat{i})} - \mathbf{y}_{(C,\hat{i})} \right\|_2^2 \end{aligned} \quad (5.3)$$

using a step-size parameter $\rho > 0$. ADMM finds a saddle point of $\mathcal{L}(\mathbf{y}_L, \boldsymbol{\alpha}, \mathbf{y})$ by updating the three blocks of variables at each iteration t :

$$\boldsymbol{\alpha}_{\hat{i}}^t \leftarrow \boldsymbol{\alpha}_{\hat{i}}^{t-1} + \rho \left(\mathbf{y}_{(L,\hat{i})}^{t-1} - \mathbf{y}_{(C,\hat{i})}^{t-1} \right) \quad \forall \hat{i} = 1, \dots, m+r \quad (5.4)$$

$$\mathbf{y}_L^t \leftarrow \arg \min_{\mathbf{y}_L} \mathcal{L}(\mathbf{y}_L, \boldsymbol{\alpha}^t, \mathbf{y}^{t-1}) \quad (5.5)$$

$$\mathbf{y}^t \leftarrow \arg \min_{\mathbf{y}} \mathcal{L}(\mathbf{y}_L^t, \boldsymbol{\alpha}^t, \mathbf{y}) \quad (5.6)$$

The ADMM updates ensure that \mathbf{y} converges to the global optimum \mathbf{y}^* , the MAP state of $P(\mathbf{y}|\mathbf{x})$, assuming that there exists a feasible assignment to \mathbf{y} . We check convergence using the criteria suggested by Boyd et al. (2011), measuring the primal and dual residuals at the end of iteration t

$$\|\bar{\mathbf{r}}^t\|_2 \triangleq \left(\sum_{\hat{i}=1}^{m+r} \left\| \mathbf{y}_{(L,\hat{i})}^t - \mathbf{y}_{(C,\hat{i})}^t \right\|_2^2 \right)^{\frac{1}{2}} \quad \|\bar{\mathbf{s}}^t\|_2 \triangleq \rho \left(\sum_{i=1}^n \mathcal{K}_i (y_i^t - y_i^{t-1})^2 \right)^{\frac{1}{2}} \quad (5.7)$$

where \mathcal{K}_i is the number of copies made of the variable y_i , i.e., the number of different potentials and constraints in which the variable participates. The updates are

terminated when both of the following conditions are satisfied

$$\|\bar{\mathbf{r}}^t\|_2 \leq \epsilon^{\text{abs}} \sqrt{\sum_{i=1}^n \mathcal{K}_i} + \epsilon^{\text{rel}} \max \left\{ \left(\sum_{i=1}^{m+r} \|\mathbf{y}_{(L,i)}^t\|_2^2 \right)^{\frac{1}{2}}, \left(\sum_{i=1}^n \mathcal{K}_i (y_i^t)^2 \right)^{\frac{1}{2}} \right\} \quad (5.8)$$

$$\|\bar{\mathbf{s}}^t\|_2 \leq \epsilon^{\text{abs}} \sqrt{\sum_{i=1}^n \mathcal{K}_i} + \epsilon^{\text{rel}} \left(\sum_{i=1}^{m+r} \|\boldsymbol{\alpha}_i^t\|_2^2 \right)^{\frac{1}{2}} \quad (5.9)$$

using convergence parameters ϵ^{abs} and ϵ^{rel} .

5.2 Block Updates

We now describe how to implement the ADMM block updates (5.4), (5.5), and (5.6).

Updating the Lagrange multipliers $\boldsymbol{\alpha}$ is a simple step in the gradient direction (5.4).

Updating the local copies \mathbf{y}_L (5.5) decomposes over each potential and constraint in the HL-MRF. For the variables $\mathbf{y}_{(L,j)}$ for each potential ϕ_j , this requires independently optimizing the weighted potential plus a squared norm:

$$\arg \min_{\mathbf{y}_{(L,j)}} w_j \left(\max \{ \ell_j(\mathbf{y}_{(L,j)}, \mathbf{x}), 0 \} \right)^{p_j} + \frac{\rho}{2} \left\| \mathbf{y}_{(L,j)} - \mathbf{y}_{(C,j)} + \frac{1}{\rho} \boldsymbol{\alpha}_j \right\|_2^2. \quad (5.10)$$

Although this optimization problem is convex, the presence of the hinge function complicates it. It could be solved in principle with an iterative method, such as an interior-point method, but this would become very expensive over many ADMM updates. Fortunately, we can reduce the problem to checking several cases and find solutions much more quickly.

There are three cases for $\mathbf{y}_{(L,j)}^*$, the optimizer of problem (5.10), which corre-

spond to the three regions in which the solution must lie: (1) the region $\ell(\mathbf{y}_{(L,j)}, \mathbf{x}) < 0$, (2) the region $\ell(\mathbf{y}_{(L,j)}, \mathbf{x}) > 0$, and (3) the region $\ell(\mathbf{y}_{(L,j)}, \mathbf{x}) = 0$. We check each case by replacing the potential with its value on the corresponding region, optimizing, and checking if the optimizer is in the correct region. We check the first case by replacing the potential ϕ_j with zero. Then, the optimizer of the modified problem is $\mathbf{y}_{(C,j)} - \boldsymbol{\alpha}_j/\rho$. If $\ell_j(\mathbf{y}_{(C,j)} - \boldsymbol{\alpha}_j/\rho, \mathbf{x}) \leq 0$, then $\mathbf{y}_{(L,j)}^* = \mathbf{y}_{(C,j)} - \boldsymbol{\alpha}_j/\rho$, because it optimizes both the potential and the squared norm independently. If instead $\ell_j(\mathbf{y}_{(C,j)} - \boldsymbol{\alpha}_j/\rho, \mathbf{x}) > 0$, then we can conclude that $\ell_j(\mathbf{y}_{(L,j)}^*, \mathbf{x}) \geq 0$, leading to one of the next two cases.

In the second case, we replace the maximum term with the inner linear function. Then the optimizer of the modified problem is found by taking the gradient of the objective with respect to $\mathbf{y}_{(L,j)}$, setting the gradient equal to the zero vector, and solving for $\mathbf{y}_{(L,j)}$. In other words, the optimizer is the solution for $\mathbf{y}_{(L,j)}$ to the equation

$$g\nabla_{\mathbf{y}_{(L,j)}} \left[w_j (\ell_j(\mathbf{y}_{(L,j)}, \mathbf{x}))^{p_j} + \frac{\rho}{2} \left\| \mathbf{y}_{(L,j)} - \mathbf{y}_{(C,j)} + \frac{1}{\rho} \boldsymbol{\alpha}_j \right\|_2^2 \right] = \mathbf{0} . \quad (5.11)$$

This is a simple system of linear equations. If $p_j = 1$, then the coefficient matrix is diagonal and trivially solved by inspection. If $p_j = 2$, then the coefficient matrix is symmetric and positive definite, and the system can be solved via Cholesky decomposition. (Since the potentials of an HL-MRF often have shared structures, perhaps templated by a PSL program, the Cholesky decompositions can be cached and shared among potentials for improved performance.) Let $\mathbf{y}'_{(L,j)}$ be the optimizer

of the modified problem, i.e., the solution to equation (5.11). If $\ell_j(\mathbf{y}'_{(L,j)}, \mathbf{x}) \geq 0$, then $\mathbf{y}^*_{(L,j)} = \mathbf{y}'_{(L,j)}$ because we know the solution lies in the region $\ell_j(\mathbf{y}_{(L,j)}, \mathbf{x}) \geq 0$ and the objective of problem (5.10) and the modified objective are equal on that region. In fact, if $p_j = 2$, then $\ell_j(\mathbf{y}'_{(L,j)}, \mathbf{x}) \geq 0$ whenever $\ell_j(\mathbf{y}_{(C,j)} - \boldsymbol{\alpha}_j/\rho, \mathbf{x}) \geq 0$, because the modified term is symmetric about the line $\ell_j(\mathbf{y}_{(L,j)}, \mathbf{x}) = 0$. We therefore will only reach the following third case when $p_j = 1$. If $\ell_j(\mathbf{y}_{(C,j)} - \boldsymbol{\alpha}_j/\rho, \mathbf{x}) > 0$ and $\ell_j(\mathbf{y}'_{(L,j)}, \mathbf{x}) < 0$, then we can conclude that $\mathbf{y}^*_{(L,j)}$ is the projection of $\mathbf{y}_{(C,j)} - \boldsymbol{\alpha}_j/\rho$ onto the hyperplane $c_k(\mathbf{y}_{(L,j)}, \mathbf{x}) = 0$. This constraint must be active because it is violated by the optimizers of both modified objectives (Martins et al., 2015, Lemma 17). Since the potential has a value of zero whenever the constraint is active, solving problem (5.10) reduces to the projection operation.

For the local copies $\mathbf{y}_{(L,k+m)}$ for each constraint c_k , the subproblem is easier:

$$\arg \min_{\mathbf{y}_{(L,k+m)}} I_k [c_k(\mathbf{y}_{(L,k+m)}, \mathbf{x})] + \frac{\rho}{2} \left\| \mathbf{y}_{(L,k+m)} - \mathbf{y}_{(C,k+m)} + \frac{1}{\rho} \boldsymbol{\alpha}_{k+m} \right\|_2^2. \quad (5.12)$$

Whether c_k is an equality or inequality constraint, the solution is the projection of $\mathbf{y}_{(C,k+m)} - \boldsymbol{\alpha}_{k+m}/\rho$ to the feasible set defined by the constraint. If c_k is an equality constraint, i.e., $k \in \mathcal{E}$, then the optimizer $\mathbf{y}^*_{(L,k+m)}$ is the projection of $\mathbf{y}_{(C,k+m)} - \boldsymbol{\alpha}_{k+m}/\rho$ onto $c_k(\mathbf{y}_{(L,k+m)}, \mathbf{x}) = 0$. If, on the other hand, c_k is an inequality constraint, i.e., $k \in \mathcal{I}$, then there are two cases. First, if $c_k(\mathbf{y}_{(C,k+m)} - \boldsymbol{\alpha}_{k+m}/\rho, \mathbf{x}) \leq 0$, then the solution is simply $\mathbf{y}_{(C,k+m)} - \boldsymbol{\alpha}_{k+m}/\rho$. Otherwise, it is again the projection onto $c_k(\mathbf{y}_{(L,k+m)}, \mathbf{x}) = 0$.

To update the variables \mathbf{y} (5.6), we solve the optimization

$$\arg \min_{\mathbf{y}} \sum_{i=1}^n I_{[0,1]} [y_i] + \frac{\rho}{2} \sum_{\hat{i}=1}^{m+r} \left\| \mathbf{y}_{(L,\hat{i})} - \mathbf{y}_{(C,\hat{i})} + \frac{1}{\rho} \boldsymbol{\alpha}_{\hat{i}} \right\|_2^2. \quad (5.13)$$

The optimizer is the state in which y_i is set to the average of its corresponding local copies added with their corresponding Lagrange multipliers divided by the step size ρ , and then clipped to the $[0, 1]$ interval. More formally, let $\mathbf{copies}(y_i)$ be the set of local copies y_c of y_i , each with a corresponding Lagrange multiplier α_c . Then, we update each y_i using

$$y_i \leftarrow \frac{1}{|\mathbf{copies}(y_i)|} \sum_{y_c \in \mathbf{copies}(y_i)} \left(y_c + \frac{\alpha_c}{\rho} \right) \quad (5.14)$$

and clip the result to $[0, 1]$. Specifically, if, after update (5.14), $y_i > 1$, then we set y_i to 1 and likewise set it to 0 if $y_i < 0$.

Algorithm 1 gives the complete pseudocode for MAP inference. It starts by initializing local copies of the variables that appear in each potential and constraint, along with a corresponding Lagrange multiplier for each copy. Then, until convergence, it iteratively performs the updates (5.4), (5.5), and (5.6). In the pseudocode, we have interleaved updates (5.4) and (5.5), updating both the Lagrange multipliers $\boldsymbol{\alpha}_{\hat{i}}$ and the local copies $\mathbf{y}_{(L,\hat{i})}$ together for each subproblem, because they are local operations that do not depend on other variables once \mathbf{y} is updated in the previous iteration. This reveals another advantage of our inference algorithm: it is very easy to parallelize. The updates (5.4) and (5.5) can be performed in parallel, the results

Algorithm 1 MAP Inference for HL-MRFs

Input: HL-MRF $P(\mathbf{y}|\mathbf{x})$, $\rho > 0$

Initialize $\mathbf{y}_{(L,j)}$ as local copies of variables $\mathbf{y}_{(C,j)}$ that are in ϕ_j , $j = 1, \dots, m$

Initialize $\mathbf{y}_{(L,k+m)}$ as local copies of variables $\mathbf{y}_{(C,k+m)}$ that are in c_k , $k = 1, \dots, r$

Initialize Lagrange multipliers $\alpha_{\hat{i}}$ corresponding to copies $\mathbf{y}_{(L,\hat{i})}$, $\hat{i} = 1, \dots, m+r$

while not converged **do**

for $j = 1, \dots, m$ **do**

$$\alpha_j \leftarrow \alpha_j + \rho(\mathbf{y}_{(L,j)} - \mathbf{y}_{(C,j)})$$

$$\mathbf{y}_{(L,j)} \leftarrow \mathbf{y}_{(C,j)} - \frac{1}{\rho}\alpha_j$$

if $\ell_j(\mathbf{y}_{(L,j)}, \mathbf{x}) > 0$ **then**

$$\mathbf{y}_{(L,j)} \leftarrow \arg \min_{\mathbf{y}_{(L,j)}} w_j \left(\ell_j(\mathbf{y}_{(L,j)}, \mathbf{x}) \right)^{p_j} + \frac{\rho}{2} \left\| \mathbf{y}_{(L,j)} - \mathbf{y}_{(C,j)} + \frac{1}{\rho}\alpha_j \right\|_2^2$$

if $\ell_j(\mathbf{y}_{(L,j)}, \mathbf{x}) < 0$ **then**

$$\mathbf{y}_{(L,j)} \leftarrow \text{Proj}_{\ell_j=0}(\mathbf{y}_{(C,j)} - \frac{1}{\rho}\alpha_j)$$

end if

end if

end for

for $k = 1, \dots, r$ **do**

$$\alpha_{k+m} \leftarrow \alpha_{k+m} + \rho(\mathbf{y}_{(L,k+m)} - \mathbf{y}_{(C,k+m)})$$

$$\mathbf{y}_{(L,k+m)} \leftarrow \text{Proj}_{c_k}(\mathbf{y}_{(C,k+m)} - \frac{1}{\rho}\alpha_{k+m})$$

end for

for $i = 1, \dots, n$ **do**

$$y_i \leftarrow \frac{1}{|\text{copies}(y_i)|} \sum_{y_c \in \text{copies}(y_i)} \left(y_c + \frac{\alpha_c}{\rho} \right)$$

 Clip y_i to $[0,1]$

end for

end while

gathered, update (5.6) performed, and the updated \mathbf{y} broadcast back to the sub-problems. Parallelization makes our MAP inference algorithm even faster and more scalable.

5.3 Lazy MAP Inference

One interesting and useful property of HL-MRFs is that it is not always necessary to completely materialize the distribution in order to find a MAP state. Consider a subset $\hat{\phi}$ of the index set $\{1, \dots, m\}$ of the potentials ϕ . Observe that if a feasible assignment to \mathbf{y} minimizes

$$\sum_{j \in \hat{\phi}} w_j \phi_j(\mathbf{y}, \mathbf{x}) \tag{5.15}$$

and $\phi_j(\mathbf{y}, \mathbf{x}) = 0, \forall j \notin \hat{\phi}$, then that assignment must be a MAP state because 0 is the global minimum for any potential. Therefore, if we can identify a set of potentials that is small, such that all the other potentials are 0 in a MAP state, then we can perform MAP inference in a reduced amount of time. Of course, identifying this set is as hard as MAP inference itself, but we can iteratively grow the set by starting with an initial set, performing inference over the current set, adding any potentials that have nonzero values, and repeating.

Since the lazy inference procedure requires that the assignment be feasible, there are two ways to handle any constraints in the HL-MRF. One is to include all constraints in the inference problem from the beginning. This will ensure feasibility but the idea of lazy grounding can also be extended to constraints to improve performance further. Just as we check if potentials are unsatisfied, i.e., nonzero,

we can also check if constraints are unsatisfied, i.e., violated. So the algorithm now iteratively grows the set of active potentials and active constraints, adding any that are unsatisfied until the MAP state of the HL-MRF defined by the active potentials and constraints is also a feasible MAP state of the true HL-MRF.

The efficiency of lazy MAP inference can be improved heuristically by not adding all unsatisfied potentials and constraints, but instead only adding those that are unsatisfied by some threshold. Although the results are no longer guaranteed to be correct, this can decrease computational cost significantly. Better understanding the effects of this heuristic and perhaps even bounding the result error when possible is an important direction for future work.

5.4 Evaluation of MAP Inference

In this section we evaluate the empirical performance of our MAP inference algorithm. We first evaluate its scalability in Section 5.4.1, comparing its running times against those of MOSEK¹ a commercially available convex optimization toolkit which uses interior-point methods (IPMs). We confirm the results of Yanover et al. (2006) that IPMs do not scale well to large-scale SP problems, and we show that our MAP inference algorithm scales very well. In fact, we observe that it scales linearly in practice with the number of potentials and constraints in the HL-MRF. In Section 5.4.2 we show that our algorithm is also an excellent approach to approximating MAP inference in discrete MRFs defined using logic by solving the local consistency relaxation (LCR) in its compact, primal form and applying the rounding guarantees

¹<http://www.mosek.com>

of Goemans and Williamson (1994) as discussed in Section 3.1. We show that it can significantly outperform MPLP (Globerson and Jaakkola, 2007), a leading approach to solving the LCR. The rounding guarantees cannot be applied to MPLP because it approximates MAP via a dual to the LCR objective. Together, these results show that HL-MRFs are extremely scalable and can also accurately approximate discrete inference problems.

5.4.1 Scalability

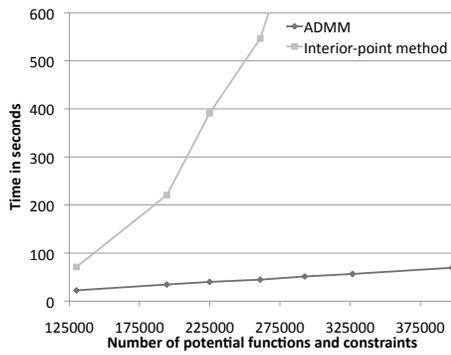
We evaluate the scalability of MAP inference by generating social networks of varying sizes, constructing HL-MRFs with them, and measuring the running time required to find a MAP state. We compare our algorithm to MOSEK’s IPM. The social networks we generate are designed to be representative of common social-network analysis tasks. We generate networks of users that are connected by different types of relationships, such as friendship and marriage, and our goal is to predict the political preferences, e.g., liberal or conservative, of each user. We also assume that we have local information about each user, which is common, representing demographic information and other indicators that are features.

We generate the social networks using power-law distributions according to a procedure described by Broecheler et al. (2010b). For a target number of users N , in-degrees and out-degrees d for each edge type are sampled from the power-law distribution $D(k) \equiv \alpha k^{-\gamma}$. Incoming and outgoing edges of the same type are then matched randomly to create edges until no more matches are possible. The

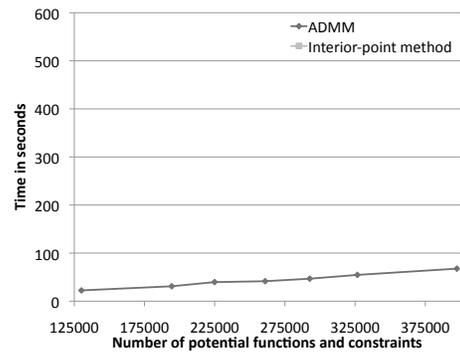
number of users is initially the target number plus the expected number of users with zero edges, and then users without any edges are removed. We use six edge types with various parameters to represent relationships in social networks with different combinations of abundance and exclusivity, choosing γ between 2 and 3, and α between 0 and 1, as suggested by Broecheler et. al. We then annotate each vertex with a value in $[-1, 1]$ uniformly at random to represent intrinsic opinions

We generate social networks with between 22,050 and 66,150 vertices, which induce HL-MRFs with between 130,080 and 397,488 total potential functions and constraints. In all the HL-MRFs, between 83% and 85% of those totals are potential functions and between 15% and 17% are constraints. For each social network, we create both a (log) piecewise-linear HL-MRF ($p_j = 1, \forall j = 1, \dots, m$ in Definition 3) and a piecewise-quadratic one ($p_j = 2, \forall j = 1, \dots, m$). We choose $\Lambda_{opinion} = 0.5$ and choose $\Lambda_{\tau_1}, \dots, \Lambda_{\tau_6}$ between 0 and 1 to model both more and less influential relationships.

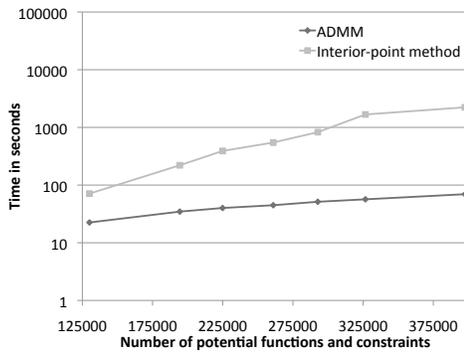
We implement ADMM in Java and compare with MOSEK’s interior-point method by encoding the entire MPE problem as a linear program or a second-order cone program as appropriate and passing the encoded problem via the Java native interface wrapper. All experiments are performed on a single machine with a 4-core 3.4 GHz Intel Core i7-3770 processor with 32GB of RAM. Each optimizer used a single thread, and all results are averaged over 3 runs. All differences between ADMM and the interior-point method on piecewise-linear problems are significant at $p = 0.0005$ using a paired t-test. All differences between ADMM and the interior-point method on piecewise quadratic problems are significant at $p = 0.0000005$.



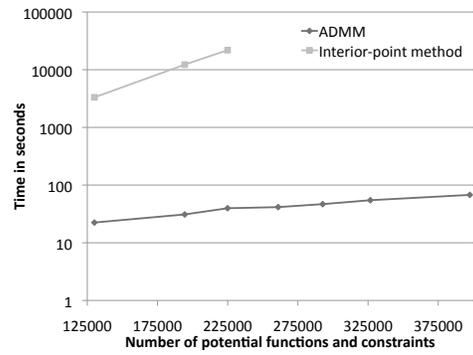
(a) Linear MAP problems



(b) Quadratic MAP problems



(c) Linear MAP problems (log scale)



(d) Quadratic MAP problems (log scale)

Figure 5.1: Average running times to find a MAP state for HL-MRFs.

We first evaluate the scalability of ADMM when solving piecewise-linear MAP problems and compare with MOSEK’s interior-point method. Figures 5.1a (normal scale) and 5.1c (log scale) show the results. The running time of the interior-point method quickly explodes as the problem size increases. The IPM’s average running time on the largest problem is about 2,200 seconds (37 minutes). This demonstrates the limited scalability of the interior-point method. In contrast, ADMM displays excellent scalability. The average running time on the largest problem is about 70 seconds. Further, the running time grows linearly in the number of potential functions and constraints in the HL-MRF, i.e., the number of subproblems that must be solved at each iteration. The line of best fit for all runs on all sizes has $R^2 = 0.9972$. Combined with Figure 5.1a, this shows that ADMM scales linearly with increasing problem size in this experiment. We emphasize that the implementation of ADMM is research code written in Java and the interior-point method is a commercial package running as native code.

We then evaluate the scalability of ADMM when solving piecewise-quadratic MAP problem and again compare with MOSEK. Figures 5.1b (normal scale) and 5.1d (log scale) show the results. Again, the running time of the interior-point method quickly explodes. We can only test it on the three smallest problems, the largest of which took an average of about 21,900 seconds to solve (over 6 hours). ADMM again scales linearly to the problem ($R^2 = 0.9854$). It is just as fast for quadratic problems as linear ones, taking average of about 70 seconds on the largest problem. The dramatic differences in running times illustrate the superior utility of our ADMM-based algorithm for these problems.

One of the advantages of interior-point methods is great numerical stability and accuracy, Consensus optimization, which treats both objective terms and constraints as subproblems, often returns points that are only optimal and feasible to moderate precision for non-trivially constrained problems (Boyd et al., 2011). Although this is often acceptable, we quantify the mix of infeasibility and suboptimality by repairing the infeasibility and measuring the resulting total suboptimality. We first project the solutions returned by consensus optimization onto the feasible region, which took a negligible amount of computational time. Let p_{ADMM} be the value of the objective in Problem (5.2) at such a point and let p_{IPM} be the value of the objective at the point returned by the interior-point method. Then the relative error on that problem is $(p_{\text{ADMM}} - p_{\text{IPM}})/p_{\text{IPM}}$. The relative error was consistently small; it varied between 0.2% and 0.4%, and did not trend upward as the problem size increased. This shows that ADMM was very accurate, in addition to being dramatically more scalable.

5.4.2 Accuracy for Approximate Discrete Inference

In this section, we compare using HL-MRFs for approximating MAP inference in discrete MRFs to using coordinate descent dual decomposition (DD), a popular approach to which rounding procedures cannot be applied (because it does not find a primal solution $\boldsymbol{\mu}^*$). We show that our proposed technique of combining the rounding procedure with message-passing algorithms can significantly improve the quality of approximate inference. We refer to our technique as rounded linear

Table 5.1: Average sizes for each group of MAP tasks.

Group	Target Users	Variables	Clauses
1	10,000	10,019	214,163
2	20,000	20,037	446,109
3	30,000	30,055	685,415
4	40,000	40,073	924,082
5	50,000	50,091	1,156,125

programming or rounded LP. We compare rounded LP with MPLP (Globerson and Jaakkola, 2007), which is a state-of-the-art coordinate descent DD algorithm. Recent work, e.g., Jojic et al. (2010) and Meshi and Globerson (2011), notes that MPLP often finds the best discrete, primal solutions.

We evaluate rounded LP and MPLP on randomly generated social network analysis problems, in which the task is to predict whether users share the same political ideology, e.g., liberal or conservative. The networks are composed of upvote and downvote edges, representing whether each user liked or disliked something another user said. We assume that we have some attribute information about each user, summarized in an ideology score uniformly distributed in the $[-1, 1]$ interval. This score could be the output of a classifier or an aggregate over features. It represents *local* information about each user, which the model considers in conjunction with the structure of the interactions.

As in Section 5.4.1, we generate networks based on the procedure of Broecheler et al. (2010b). Here we let $\alpha = 3$ and $\gamma = 2.5$. We generated 25 such networks in five groups. Table 5.1 lists the groups, target numbers of users, and the average numbers of variables and clauses in the corresponding MAP tasks, which is determined by

the networks' structures.

We construct a HL-MRF and a corresponding discrete MRF for each network to model user ideology. We describe these MRFs using PSL, since the corresponding discrete MRF except that the variables are Boolean, not continuous. A predicate will represent the ideology of users. We arbitrarily associate the true state with a liberal ideology and the false state with a conservative ideology, so we define this predicate as `Liberal/1`. For each user "u_i", we define in a PSL program a specialized rule with a weight defined by that user's ideology score. If the sign of the ideology score is positive then we add the rule

$$w_{\sim[0,1]} : \text{Liberal}(\text{"u_i"})$$

and if its sign is negative we add the rule

$$w_{\sim[0,1]} : \text{!Liberal}(\text{"u_i"})$$

In either case, each clause is weighted with the magnitude of the ideology score. For upvote edges from user X1 to X2 we encode a preference for agreeing ideology with the rules

$$1.0 : \text{Upvote}(\text{U1}, \text{U2}) \ \&\& \ \text{Liberal}(\text{U1}) \ \rightarrow \ \text{Liberal}(\text{U2})$$
$$1.0 : \text{Upvote}(\text{U1}, \text{U2}) \ \&\& \ \text{Liberal}(\text{U2}) \ \rightarrow \ \text{Liberal}(\text{U1})$$

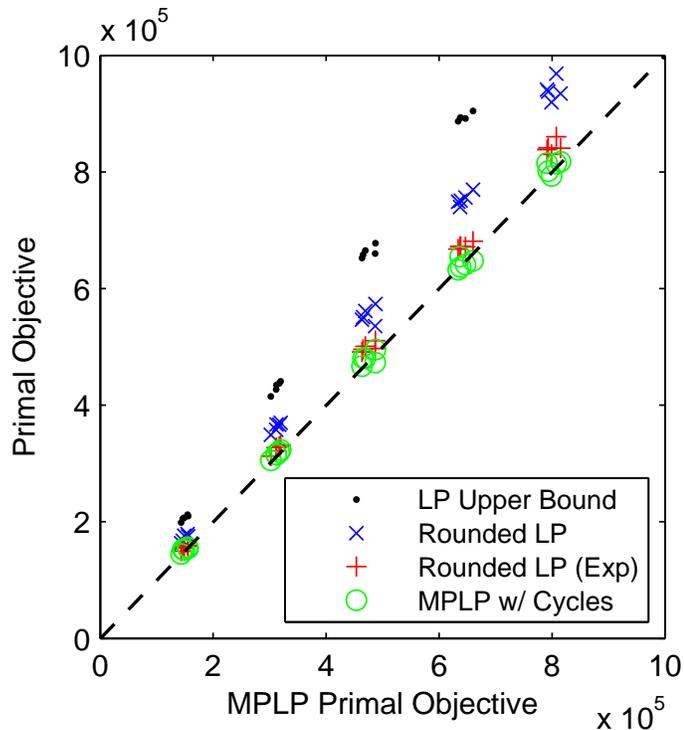


Figure 5.2: Primal objective scores relative to the score of MPLP.

and likewise for downvote edges we encode a preference for disagreeing ideology with the rules

$$1.0 : \text{Upvote}(U1, U2) \ \&\& \ \text{Liberal}(U1) \rightarrow \text{!Liberal}(U2)$$

$$1.0 : \text{Upvote}(U1, U2) \ \&\& \ \text{!Liberal}(U2) \rightarrow \text{Liberal}(U1)$$

While these models are motivated by social network analysis, they are of a similar form to many other problems and domains involving node labeling with attractive and repulsive dependencies.

For each of the 25 pairs of MRFs, we performed approximate MAP inference using rounded LP and MPLP. For rounded LP, we solved the local consistency

relaxation by performing MAP inference in the HL-MRF. We measured the initial linear program objective score (“LP Upper Bound”), which is an upper bound on any discrete primal solution, the expected score \hat{W} (3.6) using $p_i = f(\mu_i^*)$ (“Rounded LP (Exp)”), and the final score after rounding using the method of conditional probabilities (“Rounded LP”), as described in Section 3.1.1. For MPLP on the discrete MRF, we used the implementation of Globerson et al. (2012) with default settings. We measured the results of both the first-order local consistency relaxation (“MPLP”) and iterative cycle tightening (“MPLP w/ Cycles”) (Sontag et al., 2008, 2012), which searches for tighter relaxations to use. The results are summarized in Figure 5.2. All differences in scores between the ten pairs of methods, e.g., “Rounded LP (Exp)” and “MPLP w/ Cycles,” are statistically significant using a paired t-test with rejection threshold $p < 0.001$, except “MPLP” and “MPLP w/ Cycles.”

On these problems, rounded LP always outperforms MPLP. It finds solutions that are better in expectation than MPLP’s solutions, and those solutions are improved further after rounding. What makes these problems difficult is that each pair of clauses for either an upvote or downvote edge is a supermodular potential or submodular potential, respectively. The first-order local consistency relaxation would be tight for a completely supermodular problem (Wainwright and Jordan, 2008), but this mix of potentials makes the problems hard to solve. We found (in experiments not shown) that MPLP’s relative performance improves on problems that have many more supermodular potentials than submodular ones, presumably because they are very close to polynomial-time solvable problems. Cycle tightening improves the performance of MPLP, but its impact is limited because there are

so many frustrated cycles in these problems. Rounded LP is highly scalable, taking only a minute to solve problems with over one million clauses. Our experiments demonstrate tangible consequences of the approximation guarantee for rounded LP.

Chapter 6: Supervised Weight Learning

In this chapter we present three weight learning methods for HL-MRFs, each with a different objective function. The first method maximizes the likelihood of the training data. The second method maximizes the pseudolikelihood. The third method finds a large-margin solution, preferring weights that discriminate the ground truth from other states. Since weights are often shared among many potentials defined by a template, such as all the groundings of a PSL rule, we describe these learning algorithms in terms of templated HL-MRFs. We introduce some necessary notation for HL-MRF templates. Let $\mathcal{T} = (t_1, \dots, t_s)$ denote a vector of templates with associated weights $\mathbf{W} = (W_1, \dots, W_s)$. We partition the potentials by their associated templates and let t_q also denote the set of indices of the potentials defined by that template. So, $j \in t_q$ is a shorthand for saying that the potential $\phi_j(\mathbf{y}, \mathbf{x})$ was defined by template t_q . Then, we refer to the sum of the potentials defined by a template as

$$\Phi_q(\mathbf{y}, \mathbf{x}) = \sum_{j \in t_q} \phi_j(\mathbf{y}, \mathbf{x}) . \quad (6.1)$$

In the defined HL-MRF, the weight of the j -th hinge-loss potential is set to the weight of the template from which it was derived, i.e., $w_j = W_q$, for each $j \in t_q$.

Equivalently, we can rewrite the hinge-loss energy function as

$$f_{\mathbf{w}}(\mathbf{y}, \mathbf{x}) = \mathbf{W}^\top \Phi(\mathbf{y}, \mathbf{x}), \quad (6.2)$$

where $\Phi(\mathbf{y}, \mathbf{x}) = (\Phi_1(\mathbf{y}, \mathbf{x}), \dots, \Phi_s(\mathbf{y}, \mathbf{x}))$. We now describe below how to apply these learning strategies to templated HL-MRFs.

6.1 Maximum Likelihood Estimation and Structured Perceptron

The canonical approach for learning parameters \mathbf{W} is to maximize the log-likelihood of training data. The partial derivative of the log-likelihood with respect to a parameter W_q is

$$\frac{\partial \log P(\mathbf{y}|\mathbf{x})}{\partial W_q} = \mathbb{E}_{\mathbf{W}} [\Phi_q(\mathbf{y}, \mathbf{x})] - \Phi_q(\mathbf{y}, \mathbf{x}), \quad (6.3)$$

where $\mathbb{E}_{\mathbf{W}}$ is the expectation under the distribution defined by \mathbf{W} . The voted perceptron algorithm (Collins, 2002) optimizes \mathbf{W} by taking steps of fixed length in the direction of the gradient, then averaging the points after all steps. Any step that is outside the feasible region is projected back before continuing. For a smoother ascent, it is often helpful to divide the q -th component of the gradient by the number of groundings $|t_q|$ of the q -th template (Lowd and Domingos, 2007), which we do in our experiments. Computing the expectation is intractable, so we use a common approximation: the values of the potential functions at the most probable setting of \mathbf{y} with the current parameters. Using this approximation makes this approach a variant of structured perceptron.

6.2 Maximum Pseudolikelihood Estimation

Since exact maximum likelihood estimation is intractable in general, we can instead perform *maximum-pseudolikelihood estimation* (MPLE) (Besag, 1975), which maximizes the likelihood of each variable conditioned on all other variables, i.e.,

$$P^*(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^n P^*(y_i|\text{MB}(y_i), \mathbf{x}) \quad (6.4)$$

$$= \prod_{i=1}^n \frac{1}{Z_i(\mathbf{W}, \mathbf{y}, \mathbf{x})} \exp[-f_w^i(y_i, \mathbf{y}, \mathbf{x})]; \quad (6.5)$$

$$Z(w, y_i) = \int_{y_i} \exp[-f_w^i(y_i, \mathbf{y}, \mathbf{x})]; \quad (6.6)$$

$$f_w^i(y_i, \mathbf{y}, \mathbf{x}) = \sum_{j:i \in \phi_j} w_j \phi_j(\{y_i \cup \mathbf{y}_{\setminus i}\}, \mathbf{x}). \quad (6.7)$$

Here, $i \in \phi_j$ means that y_i is involved in ϕ_j , and $\text{MB}(y_i)$ denotes the *Markov blanket* of y_i —that is, the set of variables that co-occur with y_i in any potential function.

The partial derivative of the log-pseudolikelihood with respect to W_q is

$$\frac{\partial \log P^*(\mathbf{y}|\mathbf{x})}{\partial W_q} = \sum_{i=1}^n \mathbb{E}_{y_i|\text{MB}} \left[\sum_{j \in \text{t}_q: i \in \phi_j} \phi_j(\mathbf{y}, \mathbf{x}) \right] - \Phi_q(\mathbf{y}, \mathbf{x}). \quad (6.8)$$

Computing the pseudolikelihood gradient does not require inference and takes time linear in the size of \mathbf{y} . However, the integral in the above expectation does not readily admit a closed-form antiderivative, so we approximate the expectation. When a variable is unconstrained, the domain of integration is a one-dimensional interval on the real number line, so Monte Carlo integration quickly converges to an accurate

estimate of the expectation.

We can also apply MPLE when the constraints are not too interdependent. For example, for linear equality constraints over disjoint groups of variables (e.g., variable sets that must sum to 1.0), we can block-sample the constrained variables by sampling uniformly from a simplex. These types of constraints are often used to represent mutual exclusivity of classification labels. We can compute accurate estimates quickly because these blocks are typically low-dimensional.

6.3 Large-Margin Estimation

A different approach to learning drops the probabilistic interpretation of the model and views HL-MRF inference as a prediction function. Large-margin estimation (LME) shifts the goal of learning from producing accurate probabilistic models to instead producing accurate MAP predictions. The learning task is then to find the weights \mathbf{W} that provide high-accuracy structured predictions. We describe in this section a large-margin method based on the cutting-plane approach for structural support vector machines (Joachims et al., 2009).

The intuition behind large-margin structured prediction is that the ground-truth state should have energy lower than any alternate state by a large margin. In our setting, the output space is continuous, so we parameterize this margin criterion with a continuous loss function. For any valid output state $\tilde{\mathbf{y}}$, a large-margin solution should satisfy:

$$f_{\mathbf{w}}(\mathbf{y}, \mathbf{x}) \leq f_{\mathbf{w}}(\tilde{\mathbf{y}}, \mathbf{x}) - L(\mathbf{y}, \tilde{\mathbf{y}}), \quad \forall \tilde{\mathbf{y}}, \quad (6.9)$$

where the loss function $L(\mathbf{y}, \tilde{\mathbf{y}})$ measures the disagreement between a state $\tilde{\mathbf{y}}$ and the training label state \mathbf{y} . A common assumption is that the loss function decomposes over the prediction components, i.e., $L(\mathbf{y}, \tilde{\mathbf{y}}) = \sum_i L(y_i, \tilde{y}_i)$. In this work, we use the ℓ_1 distance as the loss function, so $L(\mathbf{y}, \tilde{\mathbf{y}}) = \sum_i \|y_i - \tilde{y}_i\|_1$. Since we do not expect all problems to be perfectly separable, we relax the large-margin constraint with a penalized slack ξ . We obtain a convex learning objective for a large-margin solution

$$\begin{aligned} \min_{\mathbf{W} \geq 0} \quad & \frac{1}{2} \|\mathbf{W}\|^2 + C\xi \\ \text{s.t.} \quad & \mathbf{W}^\top (\Phi(\mathbf{y}, \mathbf{x}) - \Phi(\tilde{\mathbf{y}}, \mathbf{x})) \leq -L(\mathbf{y}, \tilde{\mathbf{y}}) + \xi, \quad \forall \tilde{\mathbf{y}}, \end{aligned} \tag{6.10}$$

where $\Phi(\mathbf{y}, \mathbf{x}) = (\Phi_1(\mathbf{y}, \mathbf{x}), \dots, \Phi_s(\mathbf{y}, \mathbf{x}))$ and $C > 0$ is a user-specified parameter. This formulation is analogous to the margin-rescaling approach by Joachims et al. (2009). Though such a structured objective is natural and intuitive, its number of constraints is the cardinality of the output space, which here is infinite. Following their approach, we optimize subject to the infinite constraint set using a *cutting-plane algorithm*: we greedily grow a set K of constraints by iteratively adding the worst-violated constraint given by a *separation oracle*, then updating \mathbf{W} subject to the current constraints. The goal of the cutting-plane approach is to efficiently find the set of active constraints at the solution for the full objective, without having to enumerate the infinite inactive constraints. The worst-violated constraint is

$$\arg \min_{\tilde{\mathbf{y}}} \mathbf{W}^\top \Phi(\tilde{\mathbf{y}}, \mathbf{x}) - L(\mathbf{y}, \tilde{\mathbf{y}}). \tag{6.11}$$

The separation oracle performs loss-augmented inference by adding additional loss-

augmenting potentials to the HL-MRF. For ground truth in $\{0, 1\}$, these loss-augmenting potentials are also examples of hinge-losses, and thus adding them simply creates an augmented HL-MRF. The worst-violated constraint is then computed as standard inference on the loss-augmented HL-MRF. However, ground truth values in the interior $(0, 1)$ cause any distance-based loss to be concave, which require the separation oracle to solve a non-convex objective. For interior ground truth values, we use the *difference of convex functions algorithm* (An and Tao, 2005) to find a local optimum. Since the concave portion of the loss-augmented inference objective pivots around the ground truth value, the subgradients are 1 or -1 , depending on whether the current value is greater than the ground truth. We simply choose an initial direction for interior labels by rounding, and flip the direction of the subgradients for variables whose solution states are not in the interval corresponding to the subgradient direction until convergence.

Given a set K of constraints, we solve the SVM objective as in the primal form

$$\begin{aligned} \min_{\mathbf{W} \geq 0} \quad & \frac{1}{2} \|\mathbf{W}\|^2 + C\xi \\ \text{s.t.} \quad & K. \end{aligned} \tag{6.12}$$

We then iteratively invoke the separation oracle to find the worst-violated constraint. If this new constraint is not violated, or its violation is within numerical tolerance, we have found the max-margin solution. Otherwise, we add the new constraint to K , and repeat.

One fact of note is that the large-margin criterion always requires some slack for HL-MRFs with squared potentials. Since the squared hinge potential is quadratic

and the loss is linear, there always exists a small enough distance from the ground truth such that an absolute (i.e., linear) distance is greater than the squared distance. In these cases, the slack parameter trades off between the peakedness of the learned quadratic energy function and the margin criterion.

6.4 Evaluation of Supervised Learning

To demonstrate the flexibility and effectiveness of supervised learning with HL-MRFs, we test them on four diverse tasks: node labeling, link labeling, link prediction, and image completion. Each of these experiments represents a problem domain that is best solved with SP approaches because their dependencies are highly structural. The experiments show that HL-MRFs perform as well as or better than state-of-the-art approaches.

For these diverse tasks, we compare against a number of competing methods. For node and link labeling, we compare HL-MRFs to discrete Markov random fields (MRFs). We construct them with Markov logic networks (MLNs) Richardson and Domingos (2006), which template discrete MRFs using logical rules similarly to PSL. We perform inference in discrete MRFs using the sampling algorithm MC-SAT, and we find approximate MAP states during learning using the search algorithm MaxWalkSat Richardson and Domingos (2006). For link prediction for preference prediction, a task that is inherently continuous and nontrivial to encode in discrete logic, we compare against Bayesian probabilistic matrix factorization (BPMF) (Salakhutdinov and Mnih, 2008). Finally, for image completion, we run the same

experimental setup as Poon and Domingos (2011) and compare against the results they report, which include tests using sum product networks, deep belief networks (Hinton and Salakhutdinov, 2006), and deep Boltzmann machines (Salakhutdinov and Hinton, 2009).

We train HL-MRFs and discrete MRFs with all three learning methods: maximum likelihood estimation (MLE), maximum pseudolikelihood estimation (MPLE), and large-margin estimation (LME). When appropriate, we evaluate statistical significance using a paired t-test with rejection threshold 0.01. We describe the HL-MRFs used for our experiments using the PSL rules that define them. To investigate the differences between linear and squared potentials we use both in our experiments. HL-MRF-L refers to a model with all linear potentials and HL-MRF-Q to one with all squared potentials. When training with MLE and MPLE, we use 100 steps of voted perceptron and a step size of 1.0 (unless otherwise noted), and for LME we set $C = 0.1$. We experimented with various settings, but the scores of HL-MRFs and discrete MRFs were not sensitive to changes.

6.4.1 Node Labeling

When classifying documents, links between those documents—such as hyperlinks, citations, or co-authorship—provide extra signal beyond the local features of individual documents. Collectively predicting document classes with these links tends to improve accuracy (Sen et al., 2008). We classify documents in citation networks using data from the Cora and Citeseer scientific paper repositories. The Cora data

set contains 2,708 papers in seven categories, and 5,429 directed citation links. The Citeseer data set contains 3,312 papers in six categories, and 4,591 directed citation links. Let the predicate `Category/2` represent the category of each document and `Cites/2` represent a citation from one document to another.

The prediction task is, given a set of seed documents whose labels are observed, to infer the remaining document classes by propagating the seed information through the network. For each of 20 runs, we split the data sets 50/50 into training and testing partitions, and seed half of each set. To predict discrete categories with HL-MRFs we predict the category with the highest predicted value.

We compare HL-MRFs to discrete MRFs on this task. For prediction, we performed 2500 rounds of MC-SAT, of which 500 were burn in. We construct both using the same logical rules, which simply encode the tendency for a class to propagate across citations. For each category "C_i", we have two separate rules for each direction of citation:

$$\text{Category}(A, \text{"C_i"}) \ \&\& \ \text{Cites}(A, B) \ \rightarrow \ \text{Category}(B, \text{"C_i"})$$

$$\text{Category}(A, \text{"C_i"}) \ \&\& \ \text{Cites}(B, A) \ \rightarrow \ \text{Category}(B, \text{"C_i"})$$

We also constrain the atoms of the `Category/2` predicate to sum to 1.0 for a given document:

$$\text{Category}(D, +C) = 1.0 \ .$$

Table 6.1 lists the results of this experiment. HL-MRFs are the most accurate

Table 6.1: Average accuracy of classification by HL-MRFs and discrete MRFs. Scores statistically equivalent to the best scoring method are typed in bold.

	Citeseer	Cora
HL-MRF-Q (MLE)	0.729	0.816
HL-MRF-Q (MPLE)	0.729	0.818
HL-MRF-Q (LME)	0.683	0.789
HL-MRF-L (MLE)	0.724	0.802
HL-MRF-L (MPLE)	0.729	0.808
HL-MRF-L (LME)	0.695	0.789
MRF (MLE)	0.686	0.756
MRF (MPLE)	0.715	0.797
MRF (LME)	0.687	0.783

predictors on both data sets. Both variants of HL-MRFs are also much faster than discrete MRFs. See Table 6.3 for average inference times on five folds.

6.4.2 Link Labeling

An emerging problem in the analysis of online social networks is the task of inferring the level of trust between individuals. Predicting the strength of trust relationships can provide useful information for viral marketing, recommendation engines, and internet security. HL-MRFs with linear potentials have recently been applied by Huang et al. (2013) to this task, showing superior results with models based on sociological theory. We reproduce their experimental setup using their sample of the signed Epinions trust network, originally collected by Richardson et al. (2003), in which users indicate whether they trust or distrust other users. We perform eight-fold cross-validation. In each fold, the prediction algorithm observes the entire

Table 6.2: Average area under ROC and precision-recall curves of social-trust prediction by HL-MRFs and discrete MRFs. Scores statistically equivalent to the best scoring method by metric are typed in bold.

	ROC	P-R (+)	P-R (-)
HL-MRF-Q (MLE)	0.822	0.978	0.452
HL-MRF-Q (MPLE)	0.832	0.979	0.482
HL-MRF-Q (LME)	0.814	0.976	0.462
HL-MRF-L (MLE)	0.765	0.965	0.357
HL-MRF-L (MPLE)	0.757	0.963	0.333
HL-MRF-L (LME)	0.783	0.967	0.453
MRF (MLE)	0.655	0.942	0.270
MRF (MPLE)	0.725	0.963	0.298
MRF (LME)	0.795	0.973	0.441

unsigned social network and all but 1/8 of the trust ratings. We measure prediction accuracy on the held-out 1/8. The sampled network contains 2,000 users, with 8,675 signed links. Of these links, 7,974 are positive and only 701 are negative, making it a sparse prediction task.

We use a model based on the social theory of *structural balance*, which suggests that social structures are governed by a system that prefers triangles that are considered balanced. Balanced triangles have an odd number of positive trust relationships; thus, considering all possible directions of links that form a triad of users, there are sixteen logical implications of the form

$$\text{Trusts}(A,B) \ \&\& \ \text{Trusts}(B,C) \ \rightarrow \ \text{Trusts}(A,C)$$

Huang et al. (2013) list all sixteen of these rules, a reciprocity rule, and a prior in

their *Balance-Recip* model, which we omit to save space.

Since we expect some of these structural implications to be more or less accurate, learning weights for these rules provides better models. Again, we use these rules to define HL-MRFs and discrete MRFs, and we train them using various learning algorithms. For inference with discrete MRFs, we perform 5000 rounds of MC-SAT, of which 500 are burn in. We compute three metrics: the area under the receiver operating characteristic (ROC) curve, and the areas under the precision-recall curves for positive trust and negative trust. On all three metrics, HL-MRFs with squared potentials score significantly higher. The differences among the learning methods for squared HL-MRFs are insignificant, but the differences among the models is statistically significant for the ROC metric. For area under the precision-recall curve for positive trust, discrete MRFs trained with LME are statistically tied with the best score, and both HL-MRF-L and discrete MRFs trained with LME are statistically tied with the best area under the precision-recall curve for negative trust. The results are listed in Table 6.2.

Though the random fold splits are not the same, using the same experimental setup, Huang et al. (2013) also scored the precision-recall area for negative trust of standard trust prediction algorithms EigenTrust and TidalTrust, which scored 0.131 and 0.130, respectively. The logical models based on structural balance that we run here are significantly more accurate, and HL-MRFs more than discrete MRFs.

In addition to comparing favorably with regard to predictive accuracy, inference in HL-MRFs is also much faster than in discrete MRFs. Table 6.3 lists average inference times on five folds of three prediction tasks: Cora, Citeseer, and Epinions.

Table 6.3: Average inference times (reported in seconds) of single-threaded HL-MRFs and discrete MRFs.

	Citeseer	Cora	Epinions
HL-MRF-Q	0.42	0.70	0.32
HL-MRF-L	0.46	0.50	0.28
MRF	110.96	184.32	212.36

This illustrates an important difference between performing structured prediction via convex inference versus sampling in a discrete prediction space: using our MAP inference algorithm is much faster.

6.4.3 Link Prediction

Preference prediction is the task of inferring user attitudes (often quantified by ratings) toward a set of items. This problem is naturally structured, since a user’s preferences are often interdependent, as are an item’s ratings. *Collaborative filtering* is the task of predicting unknown ratings using only a subset of observed ratings. Methods for this task range from simple nearest-neighbor classifiers to complex latent factor models. More generally, this problem is an instance of link prediction, since the goal is to predict links indicating preference between users and content. Since preferences are ordered rather than Boolean, it is natural to represent them with the continuous variables of HL-MRFs, with higher values indicating greater preference. To illustrate the versatility of HL-MRFs, we design a simple, interpretable collaborative filtering model for predicting humor preferences. We test this model on the Jester dataset, a repository of ratings from 24,983 users on a set of

100 jokes (Goldberg et al., 2001). Each joke is rated on a scale of $[-10, +10]$, which we normalize to $[0, 1]$. We sample a random 2,000 users from the set of those who rated all 100 jokes, which we then split into 1,000 train and 1,000 test users. From each train and test matrix, we sample a random 50% to use as the observed features \mathbf{x} ; the remaining ratings are treated as the variables \mathbf{y} .

Our HL-MRF model uses an item-item similarity rule:

$$\text{SimRating}(J1, J2) \ \&\& \ \text{Likes}(U, J1) \ \rightarrow \ \text{Likes}(U, J2)$$

where $J1$ and $J2$ are jokes and U is a user; the predicate `Likes/2` indicates the degree of preference (i.e., rating value); and `SimRating/2` is a closed predicate that measures the mean-adjusted cosine similarity between the observed ratings of two jokes. We also include rules to enforce that `Likes(U, J)` concentrates around the observed average rating of user U (represented with the predicate `AvgUserRating/1`) and item J (represented with the predicate `AvgJokeRating/1`), and the global average

(represented with the predicate `AvgRating/1`):

`AvgUserRating(U) -> Likes(U, J)`

`Likes(U, J) -> AvgUserRating(U)`

`AvgJokeRating(J) -> Likes(U, J)`

`Likes(U, J) -> AvgJokeRating(J)`

`AvgRating("constant") -> Likes(U, J)`

`Likes(U, J) -> AvgRating("constant")`

`AvgRating("constant")` takes a placeholder constant as an argument, since there is only one grounding of it for the entire HL-MRF. Again, all three of these predicates are closed and computed using averages of observed ratings. In all cases, the observed ratings are taken only from the training data for learning (to avoid leaking information about the test data) and only from the test data during testing.

We compare our HL-MRF model to a current state-of-the-art latent factors model, *Bayesian probabilistic matrix factorization* (BPMF) (Salakhutdinov and Mnih, 2008). BPMF is a fully Bayesian treatment and, as such, is considered “parameter-free;” the only parameter that must be specified is the rank of the decomposition. Based on settings used by Xiong et al. (2010), we set the rank of the decomposition to 30 and use 100 iterations of burn in and 100 iterations of sampling. For our experiments, we use the code of Xiong et al. (2010). Since BPMF does not train a model, we allow BPMF to use all of the training matrix during the prediction phase.

Table 6.4: Normalized mean squared/absolute errors (NMSE/NMAE) for preference prediction using the Jester dataset. The lowest errors are typed in bold.

	NMSE	NMAE
HL-MRF-Q (MLE)	0.0554	0.1974
HL-MRF-Q (MPLE)	0.0549	0.1953
HL-MRF-Q (LME)	0.0738	0.2297
HL-MRF-L (MLE)	0.0578	0.2021
HL-MRF-L (MPLE)	0.0535	0.1885
HL-MRF-L (LME)	0.0544	0.1875
BPMF	0.0501	0.1832

Table 6.5: Mean squared errors per pixel for image completion. HL-MRFs produce the most accurate completions on the Caltech101 and the left-half Olivetti faces, and only sum-product networks produce better completions on Olivetti bottom-half faces. Scores for other methods are taken from Poon and Domingos (2011).

	HL-MRF-Q (MLE)	SPN	DBM	DBN	PCA	NN
Caltech-Left	1741	1815	2998	4960	2851	2327
Caltech-Bottom	1910	1924	2656	3447	1944	2575
Olivetti-Left	927	942	1866	2386	1076	1527
Olivetti-Bottom	1226	918	2401	1931	1265	1793

Table 6.4 lists the normalized mean squared error (NMSE) and normalized mean absolute error (NMAE), averaged over 10 random splits. Though BPMF produces the best scores, the improvement over HL-MRF-L (LME) is not significant in NMAE.

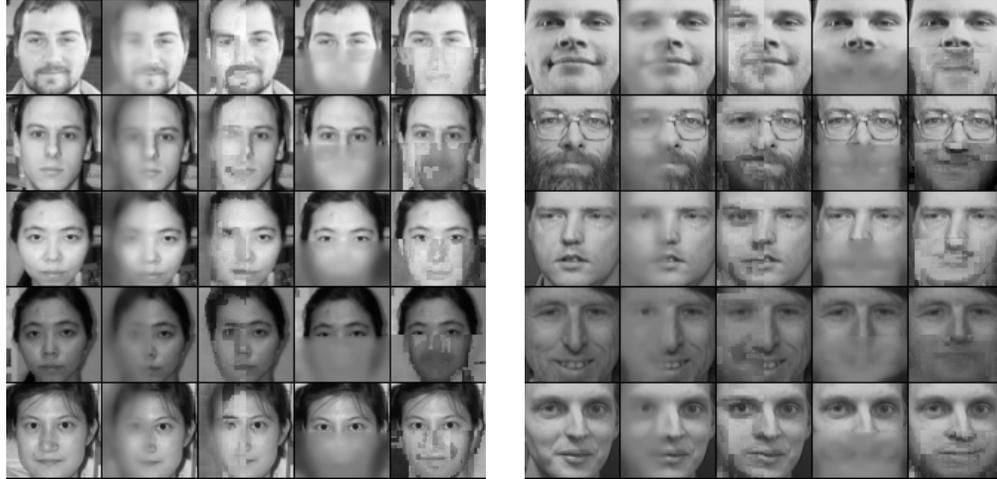


Figure 6.1: Example results on image completion of Caltech101 (left) and Olivetti (right) faces. From left to right in each column: (1) true face, left side predictions by (2) HL-MRFs and (3) SPNs, and bottom half predictions by (4) HL-MRFs and (5) SPNs. SPN completions are downloaded from Poon and Domingos (2011).

6.4.4 Image Completion

Digital image completion requires models that understand how pixels relate to each other, such that when some pixels are unobserved, the model can infer their values from parts of the image that are observed. We construct pixel-grid HL-MRFs for image completion. We test these models using the experimental setup of Poon and Domingos (2011): we reconstruct images from the Olivetti face data set and the Caltech101 face category. The Olivetti data set contains 400 images, 64 pixels wide and tall, and the Caltech101 face category contains 435 examples of faces, which we crop to the center 64 by 64 patch, as was done by Poon and Domingos (2011). Following their experimental setup, we hold out the last fifty images and predict either the left half of the image or the bottom half.

The HL-MRFs in this experiment are much more complex than the ones in

our other experiments because we allow each pixel to have its own weight for the following rules, which encode agreement or disagreement between neighboring pixels:

```
Bright("P_ij", I) && North("P_ij", Q) -> Bright(Q, I)
Bright("P_ij", I) && North("P_ij", Q) -> !Bright(Q, I)
!Bright("P_ij", I) && North("P_ij", Q) -> Bright(Q, I)
!Bright("P_ij", I) && North("P_ij", Q) -> !Bright(Q, I)
```

where $\text{Bright}(\text{"P_ij"}, I)$ is the normalized brightness of pixel "P_ij" in image I, and $\text{North}(\text{"P_ij"}, Q)$ indicates that Q is the north neighbor of "P_ij". We similarly include analogous rules for the south, east, and west neighbors, as well as the pixels mirrored across the horizontal and vertical axes. This setup results in up to 24 rules per pixel, (boundary pixels may not have north, south, east, or west neighbors) which, in a 64 by 64 image, produces 80,896 PSL rules.

We train these HL-MRFs using MLE with a 5.0 step size on the first 200 images of each data set and test on the last fifty. For training, we maximize the data log-likelihood of uniformly random held-out pixels for each training image, allowing for generalization throughout the image. Table 6.5 lists our results and others reported by Poon and Domingos (2011) for sum-product networks (SPN), deep Boltzmann machines (DBM), deep belief networks (DBN), principal component analysis (PCA), and nearest neighbor (NN). HL-MRFs produce the best mean squared error on the left- and bottom-half settings for the Caltech101 set and the left-half setting in the Olivetti set. Only sum product networks produce lower error on the Olivetti bottom-

half faces. Some reconstructed faces are displayed in Figure 6.1, where the shallow, pixel-based HL-MRFs produce comparably convincing images to sum-product networks, especially in the left-half setting, where HL-MRFs can learn which pixels are likely to mimic their horizontal mirror. While neither method is particularly good at reconstructing the bottom half of faces, the qualitative difference between the deep SPN and the shallow HL-MRF completions is that SPNs seem to hallucinate different faces, often with some artifacts, while HL-MRFs predict blurry shapes roughly the same pixel intensity as the observed, top half of the face. The tendency to better match pixel intensity helps HL-MRFs score better quantitatively on the Caltech101 faces, where the lighting conditions are more varied than in Olivetti faces.

Training and predicting with these HL-MRFs takes little time. In our experiments, training each model takes about 45 minutes on a 12-core machine, while predicting takes under a second per image. While Poon and Domingos (2011) report faster training with SPNs, both HL-MRFs and SPNs clearly belong to a class of faster models when compared to DBNs and DBMs, which can take days to train on modern hardware.

Chapter 7: Learning with Latent Variables

Latent variables can capture structure in complicated domains and have been used extensively in social and biological network analysis, Web analytics, computer vision, and many other domains that study large-scale, structured data. However, including latent variables sacrifices scalability for expressiveness because the values of latent variables are—by definition—unknown. Algorithms for learning with latent variables often require repeated inference to iteratively update parameters, and each inference alone can be expensive for a large model. For example, inference methods like Gibbs sampling and belief propagation require many iterations to converge, and learning methods like EM alternate between fully inferring latent variable values and updating parameters.

Latent variables are particularly valuable in rich, structured models, but the computational costs become even more challenging. Our contribution is a new learning framework for rich, structured, continuous latent-variable models that addresses this computational bottleneck.

Overcoming the need for repeated inference requires contending with challenges that arise from a continuous representation, including the need for efficient alternatives to representing distributions over uncountable state spaces and eval-

uating irreducible integrals. For fully-supervised learning, large-margin methods can use the dual of loss-augmented inference to form a joint convex minimization (Taskar et al., 2005; Meshi et al., 2010). Schwing et al. (2012a) extended this idea to latent-variable learning for discrete MRFs, using a method specifically formulated to pass messages corresponding to the discrete states of the variables. While these methods are incompatible with continuous models, dualization is also a key to faster training of continuous models with latent variables.

In Section 7.2, we propose *paired-dual learning*, a framework that quickly trains HL-MRFs with latent variables by avoiding repeated inferences. Traditional methods for learning with latent variables require repeated inferences for two distributions to compute gradients. The unobserved random variables are grouped into two sets, those with training labels and those without, i.e., the latent variables. One distribution is joint over the labeled variables and the latent variables, and the other is over the latent variables conditioned on the labels. Paired-dual learning uses an equivalent variational learning objective that substitutes dual problems for the two corresponding inference problems, augmented with entropy surrogates to make the learning problem well-formed. We describe how to design suitable entropy surrogates that retain the useful properties of entropy while still admitting fast HL-MRF inference. We can therefore compute the gradient of the paired-dual learning objective with respect to the parameters using the intermediate states of inference, enabling a fast, block-coordinate joint optimization.

We show in Section 7.4 that paired-dual learning drastically reduces the time required for learning without sacrificing accuracy on three real-world problems:

social-group detection, trust prediction in social networks, and image reconstruction. Paired-dual learning cuts training time by as much as 90%, often converging before traditional methods make a single update to the parameters.

7.1 Background on Variational Expectation Maximization

Paired-dual learning quickly optimizes a standard learning objective, which we review in this subsection. When learning models with latent variables, the usual goal is to maximize the marginal likelihood of the labels $\hat{\mathbf{y}}$ given observed variables \mathbf{x} , marginalizing out over all possible configurations of latent variables \mathbf{z} . For a parameter setting w and any state of the latent variables \mathbf{z} , the log marginal likelihood can be expressed as a log ratio of joint and conditional likelihoods, which simplifies to the difference of two normalizing partition functions:

$$\log P(\hat{\mathbf{y}}|\mathbf{x}; \mathbf{w}) = \log P(\hat{\mathbf{y}}, \mathbf{z}|\mathbf{x}; \mathbf{w}) - \log P(\mathbf{z}|\mathbf{x}, \hat{\mathbf{y}}; \mathbf{w}) \quad (7.1)$$

$$\begin{aligned} &= -\mathbf{w}^\top \phi(\hat{\mathbf{y}}, \mathbf{z}, \mathbf{x}) - \log Z(\mathbf{x}; \mathbf{w}) \\ &\quad + \mathbf{w}^\top \phi(\hat{\mathbf{y}}, \mathbf{z}, \mathbf{x}) + \log Z(\mathbf{x}, \hat{\mathbf{y}}; \mathbf{w}) \end{aligned} \quad (7.2)$$

$$= \log Z(\mathbf{x}, \hat{\mathbf{y}}; \mathbf{w}) - \log Z(\mathbf{x}; \mathbf{w}) . \quad (7.3)$$

Each of these partition functions has a variational form (Wainwright and Jordan, 2008), yielding the identity

$$\begin{aligned} \log Z(\mathbf{x}, \hat{\mathbf{y}}; \mathbf{w}) - \log Z(\mathbf{x}; \mathbf{w}) = & \min_{\rho \in \Delta(\mathbf{y}, \mathbf{z})} \max_{q \in \Delta(\mathbf{z})} \mathbb{E}_{\rho} [\mathbf{w}^{\top} \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}, \mathbf{z})] - H(\rho) \\ & - \mathbb{E}_q [\mathbf{w}^{\top} \boldsymbol{\phi}(\mathbf{x}, \hat{\mathbf{y}}, \mathbf{z})] + H(q) , \end{aligned} \quad (7.4)$$

where ρ is a joint distribution over the \mathbf{y} and \mathbf{z} variables from the space of all joint distributions $\Delta(\mathbf{y}, \mathbf{z})$, q is a conditional distribution over the \mathbf{z} variables from the space of all conditional distributions $\Delta(\mathbf{z})$, and H is the entropy.

Using the variational form, Equation 7.4, regularized maximum likelihood is the following saddle-point optimization:

$$\begin{aligned} \arg \min_{\mathbf{w}} \min_{\rho \in \Delta(\mathbf{y}, \mathbf{z})} \max_{q \in \Delta(\mathbf{z})} & \frac{\lambda}{2} \|\mathbf{w}\|^2 + \mathbb{E}_{\rho} [\mathbf{w}^{\top} \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}, \mathbf{z})] - H(\rho) \\ & - \mathbb{E}_q [\mathbf{w}^{\top} \boldsymbol{\phi}(\mathbf{x}, \hat{\mathbf{y}}, \mathbf{z})] + H(q) \end{aligned} \quad (7.5)$$

where $\lambda \geq 0$ is a tunable regularization parameter.¹ We solve the learning problem in its variational form because it enables principled approximations of intractable problems by restricting the spaces of distributions $\Delta(\mathbf{y}, \mathbf{z})$ and $\Delta(\mathbf{z})$.

A traditional approach for optimizing Equation 7.5 computes subgradients of the outer minimization over \mathbf{w} by exactly solving the inner min-max and differentiating. Another approach iteratively solves the conditional inference over \mathbf{z}' , fixes \mathbf{z}' , and solves the remaining min-max over \mathbf{w} and \mathbf{y}, \mathbf{z} as a fully-observed maximum-

¹We use L2 regularization in our derivations and experiments, but paired-dual learning is easily adapted to include any regularization function whose subdifferentials are computable.

likelihood estimation.² Each of these approaches performs a block coordinate ascent-descent that requires fully solving two (or more) inferences per iteration of the outer optimization.

The inner minimization and maximization have the same form as marginal inference. This equivalence suggests that the likelihood can be evaluated by performing these two inferences. When variational distributions ρ and q are optimized over the full class of all possible distributions, objective (7.5) exactly corresponds to the true marginal likelihood. However, it contains quantities—the expectations and the entropies—that are intractable to compute.

7.2 Paired-Dual Learning

In this section, we present paired-dual learning, a framework for training HL-MRFs with latent variables. Optimizing the variational learning objective, Equation 7.5, is intractable because the expectations and entropies are irreducible integrals. Traditional methods approximate the objective by restricting the variational distributions ρ and q to tractable families, and we adopt this approach as well. However, traditional methods fit and refit ρ and q exactly before each update of the parameters \mathbf{w} . Paired-dual learning speeds up training by interleaving updates of \mathbf{w} into dual optimizations over ρ and q . Dualizing these inference problems allows training to use the intermediate solutions produced by ADMM. To enable this interleaved joint optimization, we first construct surrogates for the entropy functions $H(\rho)$ and $H(q)$

²This strategy is equivalent to variational expectation maximization (EM), or “hard” EM if using point distributions, and it generalizes the standard approach for latent structured SVM (Yu and Joachims, 2009).

so that, when the variational families $\Delta(\mathbf{y}, \mathbf{z})$ and $\Delta(\mathbf{z})$ are restricted to point estimates, fitting the distributions ρ and q is subsumed by MAP inference, while still preserving the desired properties of entropy functions in learning. To optimize over the model parameters \mathbf{w} , we consider the ADMM duals of both variational inference problems, forming a new saddle-point objective that can be differentiated with respect to \mathbf{w} during intermediate stages of ADMM.

7.2.1 Tractable Entropy Surrogates

As with many continuous models, optimizing Equation 7.5 exactly for HL-MRFs is intractable because the expectations and the entropies are irreducible integrals. To remove this intractability, we first adopt the common approximation of restricting $\Delta(\mathbf{y}, \mathbf{z})$ and $\Delta(\mathbf{z})$ to tractable families of variational distributions. We restrict the variational families to point distributions, enabling highly scalable MAP inference techniques to optimize over them. In other words, the minimizing distribution ρ^* places all probability on the point (\mathbf{y}, \mathbf{z}) that minimizes $\mathbf{w}^\top \phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) - H(\rho)$, and q^* places all probability on the point \mathbf{z} that minimizes $\mathbf{w}^\top \phi(\mathbf{x}, \hat{\mathbf{y}}, \mathbf{z}) - H(q)$. Moreover, the entropies $H(\rho)$ and $H(q)$ are always zero for point distributions, so finding ρ^* and q^* for a particular \mathbf{w} are instances of MAP inference.

Using this approximation alone, Equation 7.5 always has a degenerate global optimum at $\mathbf{w} = \mathbf{0}$. This degeneracy reveals the importance of having nontrivial entropy terms to reward high-entropy states. To remove this degenerate solution, we need to include tractable surrogates for the entropies in Equation 7.5 that behave as

the true entropies should: biasing the objective away from the labeled state so that stronger weights are necessary to produce good predictions. Therefore, the surrogate entropy and the weight-norm regularization will have opposite effects, removing the degenerate zero solution.

We can preserve this non-degeneracy effect without complicating MAP inference by choosing hinge functions as entropy surrogates and treating them as potentials with fixed weights. For example, if a HL-MRF variable y represents the degree to which a person is in each of two latent groups—with $y = 0.0$ being completely in a group and $y = 1.0$ being completely in the other—then, the following pair of squared-hinge potentials can act as a suitable entropy surrogate for the point distribution at y :

$$-w (\max\{y, 0\}^2 + \max\{1 - y, 0\}^2) . \quad (7.6)$$

This entropy surrogate penalizes solutions where y deviates from 0.5, making the learning objective prefer models strong enough to push y towards one extreme. During learning, the associated parameter w is fixed, but during MAP inference the surrogate can be treated as another pair of hinge potentials, preserving the scalability of inference.

The function that acts as a surrogate does not need a probabilistic interpretation, and the appropriate choice of these surrogates can generalize the objectives of latent structured SVM (LSSVM) (Yu and Joachims, 2009) and variants of expectation maximization (EM). The LSSVM objective uses a loss between the current prediction \mathbf{y} and the labels $\hat{\mathbf{y}}$ as a surrogate for $H(\rho)$ and no surrogate, i.e., 0, for

$H(q)$. The ℓ_1 loss function can be represented with simple hinge functions, enabling HL-MRF inference. We discuss these connections further in Section 7.3.

Let h be any surrogate entropy of point distributions. The tractable latent variable HL-MRF learning objective is

$$\begin{aligned} \arg \min_{\mathbf{w}} \min_{\mathbf{y}, \mathbf{z}} \max_{\mathbf{z}'} \quad & \frac{\lambda}{2} \|\mathbf{w}\|^2 + \mathbf{w}^\top \phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) - h(\mathbf{y}, \mathbf{z}) \\ & - \mathbf{w}^\top \phi(\mathbf{x}, \hat{\mathbf{y}}, \mathbf{z}') + h(\hat{\mathbf{y}}, \mathbf{z}') . \end{aligned} \quad (7.7)$$

7.2.2 Joint Optimization

The traditional approaches involving repeatedly performing complete inference, i.e., finding \mathbf{y} , \mathbf{z} , and \mathbf{z}' in Equation 7.7, can be very expensive in large-scale settings. Instead, we derive a method that exploits that HL-MRF inference can be solved via ADMM. In particular, this method enables optimization using partial solutions to inference. That is, the optimization can proceed before the inference optimization completes its computation.

We form a new joint optimization by rewriting Equation 7.7 with the corresponding augmented Lagrangians used to solve the inner optimizations. Let $\mathcal{L}_{\mathbf{w}}(\mathbf{v}, \boldsymbol{\alpha}, \bar{\mathbf{v}})$ be the augmented Lagrangian for minimizing $\mathbf{w}^\top \phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) - h(\mathbf{y}, \mathbf{z})$. We subscript the augmented Lagrangian with the parameters \mathbf{w} to emphasize that it is also a function of the current parameters. Let $\mathcal{L}'_{\mathbf{w}}(\mathbf{v}', \boldsymbol{\alpha}', \bar{\mathbf{v}}')$ be the analogous augmented Lagrangian for minimizing $\mathbf{w}^\top \phi(\mathbf{x}, \hat{\mathbf{y}}, \mathbf{z}') - h(\hat{\mathbf{y}}, \mathbf{z}')$. Substituting them

into Equation 7.7, we write the equivalent paired-dual learning objective:

$$\arg \min_{\mathbf{w}} \min_{\mathbf{v}, \bar{\mathbf{v}}} \max_{\boldsymbol{\alpha}} \max_{\mathbf{v}', \bar{\mathbf{v}'}} \min_{\boldsymbol{\alpha}'} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \mathcal{L}_{\mathbf{w}}(\mathbf{v}, \boldsymbol{\alpha}, \bar{\mathbf{v}}) - \mathcal{L}'_{\mathbf{w}}(\mathbf{v}', \boldsymbol{\alpha}', \bar{\mathbf{v}'}) . \quad (7.8)$$

Since the inner optimizations are guaranteed to converge to the global optima for fixed \mathbf{w} , Equations 7.7 and 7.8 are identical. With this view, we no longer need to solve the optimizations to completion as they appear in the primal Equations 7.7. Instead, a finer-grained block-coordinate optimization over the variables that appear in the paired-dual Equation 7.8, interleaving subgradient steps over \mathbf{w} and ADMM iterations over the other variables, reaches an optimum more quickly.

This objective is non-convex, and determining whether any block-coordinate optimization scheme for it will converge is an open question. If the inner optimizations were solved to convergence between updates of \mathbf{w} , then the optimization provably converges as an instance of the concave-convex procedure (Yuille and Rangarajan, 2003), in the same manner as LSSVM (Yu and Joachims, 2009). Schwing et al. (2012a) derived a convergent algorithm for training discrete Markov random fields with latent variables that dualizes the optimization over (discrete) \mathbf{y} and \mathbf{z} and interleaves updating the corresponding dual variables and the parameters \mathbf{w} —while still solving the optimization over \mathbf{z}' to convergence at each iteration. This algorithm updates beliefs over discrete variables but is not applicable to the continuous, non-linear potentials of HL-MRFs. While no guarantees for paired-dual learning are known, it always converges in our diverse experiments (see Section 6.4).

7.2.3 Learning Algorithm

The complete learning algorithm is summarized in Algorithm 2. We first construct the augmented Lagrangian $\mathcal{L}_w(\mathbf{v}, \boldsymbol{\alpha}, \bar{\mathbf{v}})$ for MAP inference in $P(\mathbf{y}, \mathbf{z}|\mathbf{x}; w)$ and the analogous augmented Lagrangian $\mathcal{L}'_w(\mathbf{v}', \boldsymbol{\alpha}', \bar{\mathbf{v}}')$ for inference in $P(\mathbf{z}|\mathbf{x}, \hat{\mathbf{y}}; w)$, as described in Section 5. Then, at each iteration t , we first execute ADMM iterations, which update the Lagrangian $\mathcal{L}_w(\mathbf{v}, \boldsymbol{\alpha}, \bar{\mathbf{v}})$ by taking a step in the dual space over the variables $\boldsymbol{\alpha}$, then optimizing \mathbf{v} , and finally optimizing $\bar{\mathbf{v}}$. We limit ADMM to N iterations before moving on, where N is a user-specified parameter.³ In our experiments, we found that higher values result in slower training, and in Section 6.4, we discuss results that suggest setting $N = 1$, i.e., single updates of all variables, provides the best speed and accuracy.

We then update the other Lagrangian $\mathcal{L}'_w(\mathbf{v}', \boldsymbol{\alpha}', \bar{\mathbf{v}}')$. At the end of each iteration t , we update \mathbf{w} via the derivative of the joint objective, Equation 7.8. The gradients $\nabla_{\mathbf{w}}$ for \mathcal{L}_w and \mathcal{L}'_w are straightforward. The gradient for a potential ϕ is the potential function value at the current setting of the local copies \mathbf{v} and \mathbf{v}' . This computation only differs from how one computes the gradient in the primal setting in that it is evaluated for variable copies that might not agree during this intermediate stage. Since the weights \mathbf{w} do not interact with any of the dual terms in the augmented Lagrangian, these terms do not affect the gradient.

Naive interleaving of learning with inference could be implemented with early stopping and warm starting of ADMM inference. Without the paired-dual view, one

³If $\mathcal{L}_w(\mathbf{v}, \boldsymbol{\alpha}, \bar{\mathbf{v}})$ converges for the current setting of \mathbf{w} , we terminate the inner loop early. Therefore, each inner loop performs between 1 and N ADMM iterations at each outer iteration t .

Algorithm 2 Paired-Dual Learning

Input: model $P(\mathbf{y}, \mathbf{z}|\mathbf{x}; \mathbf{w})$, labeled data $\hat{\mathbf{y}}$, initial parameters \mathbf{w}

Form augmented Lagrangian $\mathcal{L}_w(\mathbf{v}, \boldsymbol{\alpha}, \bar{\mathbf{v}})$ for $\arg \min_{\mathbf{z}, \mathbf{y}} \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}, \mathbf{z}) - h(\mathbf{y}, \mathbf{z})$

Form augmented Lagrangian $\mathcal{L}'_w(\mathbf{v}', \boldsymbol{\alpha}', \bar{\mathbf{v}}')$ for $\arg \min_{\mathbf{z}'} \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}, \hat{\mathbf{y}}, \mathbf{z}') - h(\hat{\mathbf{y}}, \mathbf{z}')$

for t from 1 to T **do**

for n from 1 to N **or** until converged **do**

$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \rho \mathbf{c}(\mathbf{v}, \bar{\mathbf{v}})$

$\mathbf{v} \leftarrow \arg \min_{\mathbf{v}} \mathcal{L}_w(\mathbf{v}, \boldsymbol{\alpha}, \bar{\mathbf{v}})$

$\bar{\mathbf{v}} \leftarrow \arg \min_{\bar{\mathbf{v}}} \mathcal{L}_w(\mathbf{v}, \boldsymbol{\alpha}, \bar{\mathbf{v}})$

end for

for n from 1 to N **or** until converged **do**

$\boldsymbol{\alpha}' \leftarrow \boldsymbol{\alpha}' + \rho \mathbf{c}'(\mathbf{v}', \bar{\mathbf{v}}')$

$\mathbf{v}' \leftarrow \arg \min_{\mathbf{v}'} \mathcal{L}'_w(\mathbf{v}', \boldsymbol{\alpha}', \bar{\mathbf{v}}')$

$\bar{\mathbf{v}}' \leftarrow \arg \min_{\bar{\mathbf{v}}'} \mathcal{L}'_w(\mathbf{v}', \boldsymbol{\alpha}', \bar{\mathbf{v}}')$

end for

if $t > K$ **then**

$\nabla_w \leftarrow \nabla_w \left[\frac{\lambda}{2} \|\mathbf{w}\|^2 + \mathcal{L}_w(\mathbf{v}, \boldsymbol{\alpha}, \bar{\mathbf{v}}) - \mathcal{L}'_w(\mathbf{v}', \boldsymbol{\alpha}', \bar{\mathbf{v}}') \right]$

 Update \mathbf{w} via ∇_w

end if

end for

could use the gradient of the primal objective using the consensus variables $\bar{\mathbf{v}}$ and $\bar{\mathbf{v}}'$ (or some other estimate of the inference variables), but these gradients would not correspond to Equation 7.8, or to any principled objective function. Instead, the paired-dual learning objective enables joint optimization of a principled objective, with gradient computations no more complicated than in the primal setting.

Finally, one can “warm up” the ADMM variables by updating \mathbf{v} , $\boldsymbol{\alpha}$, $\bar{\mathbf{v}}$, \mathbf{v}' , $\boldsymbol{\alpha}'$, and $\bar{\mathbf{v}}'$ for a few iterations before beginning to update the parameters \mathbf{w} . Setting warm-up parameter K greater than zero can improve the initial search direction for \mathbf{w} by reducing the gap between the gradient

$$\nabla_{\mathbf{w}} \left[\frac{\lambda}{2} \|\mathbf{w}\|^2 + \min_{\mathbf{v}, \bar{\mathbf{v}}} \max_{\mathbf{v}', \bar{\mathbf{v}}'} \mathcal{L}_{\mathbf{w}}(\mathbf{v}, \boldsymbol{\alpha}, \bar{\mathbf{v}}) - \mathcal{L}'_{\mathbf{w}}(\mathbf{v}', \boldsymbol{\alpha}', \bar{\mathbf{v}}') \right]. \quad (7.9)$$

and the paired-dual learning approximation

$$\nabla_{\mathbf{w}} \left[\frac{\lambda}{2} \|\mathbf{w}\|^2 + \mathcal{L}_{\mathbf{w}}(\mathbf{v}, \boldsymbol{\alpha}, \bar{\mathbf{v}}) - \mathcal{L}'_{\mathbf{w}}(\mathbf{v}', \boldsymbol{\alpha}', \bar{\mathbf{v}}') \right] \quad (7.10)$$

for the initial setting of \mathbf{w} . In our experiments (Section 6.4), $K = 0$ often suffices, but for one task, using $K = 10$ produces a better start to optimization. The cost of this warmup is negligible, since learning often requires hundreds of ADMM iterations, but the benefits of taking a better initial gradient step can be significant in practice.

Variants of paired-dual learning easily fit into this framework. We can stop after a fixed number of iterations or when \mathbf{w} has converged. We can transparently

apply existing strategies for smoother gradient-based optimization, e.g., adaptive rescaling (Duchi et al., 2011) or averaging.

7.3 Related Approaches for Discrete Models

There exist many approaches to learning discrete, discriminative models with latent variables. Existing classes of probabilistic models include *hidden-unit conditional random fields* (van der Maaten et al., 2011), a class of undirected graphical models similar to linear conditional random fields, except that a latent variable mediates the interaction between each observation and target variable on the chain. This restricted structure allows the latent variables to be marginalized out during inference and learning but cannot express more complex dependencies. More expressive discriminative models have been trained via specialized inference algorithms designed for specific models (e.g., Kok and Domingos, 2007; Poon and Domingos, 2009). Another class of probabilistic models are *sum-product networks* (Poon and Domingos, 2011), or SPNs, which represent distributions as networks of sum and product operations. Interior nodes in an SPN have a natural interpretation as latent variables, and SPNs can be trained with EM.

The variational objective, Equation 7.7, relates to several important ideas in probabilistic inference and latent variable learning. For discrete MRFs, surrogates enable efficient and accurate inference (e.g., Heskes, 2006; Weiss et al., 2007; Wainwright and Jordan, 2008; Meshi et al., 2009). Especially for learning, no statistical interpretation of the surrogates is necessary. For example, using the family of point

distributions and replacing the entropy with a distance metric between the point and the labels, we obtain the objective for LSSVM (Yu and Joachims, 2009). Similarly, using point expectations and using null surrogates, i.e., $h(\rho) = 0$, the objective becomes analogous to structured perceptron (Collins, 2002; Richardson and Domingos, 2006). Lastly, using tractable families of distributions for both the expectation and the entropies makes the learning objective that of variational EM (Neal and Hinton, 1999).

Replacing inference problems with duals to speed up learning has also been explored for discrete models. For fully-supervised settings, Taskar et al. (2005) dualize the loss-augmented inference problem as part of large-margin learning, making a joint quadratic program. Meshi et al. (2010) improve on this approach to use dual decomposition for LP relaxations of inference in discrete graphical models. Schwing et al. (2012a) extend this idea to latent-variable models. By dualizing one of the two inference subroutines and passing messages corresponding to the discrete states, they speed up learning of discrete models with latent variables. Related to this line of work, Domke (2013) uses dualization as part of a technique to reduce structured prediction to non-structured logistic regression.

7.4 Evaluation of Learning with Latent Variables

In this section, we evaluate paired-dual learning by comparing it with traditional learning methods on real-world problems. We test two variants of paired-dual learning: the finest grained interleaving with only two ADMM iterations per weight up-

date ($N = 1$) and a coarser grained 20 ADMM iterations per update ($N = 10$). We compare with *primal subgradient*, which evaluates subgradients of Equation 7.7 by solving the inner optimizations to convergence ($N = \infty$), and *expectation maximization* (EM), which fits the parameters via multiple subgradient descent steps for each point estimate of the latent variables \mathbf{z}' .

We consider three problems: group detection in social media, social-trust prediction, and image completion. For each problem, we build HL-MRFs that include latent variables and surrogate entropies, run each learning algorithm, and evaluate on held-out test data. The iterations of ADMM constitute most of the computational cost during learning, so we measure the quality of the learned models as a function of the number of ADMM iterations taken during learning. Since each ADMM step is exactly the same amount of computation, regardless of the learning algorithm or the current model, the number of ADMM steps represents the computational cost, avoiding confounding factors such as differences in hardware used in these experiments. During each outer iteration of each algorithm, we store the current weights and later use these weights offline to measure the primal objective, Equation 7.7, and predictive performance on held out data. We provide high-level details on each experiment and defer additional details to the appendix.

For all four methods, we update weights using a standard subgradient descent approach for large-scale MRFs (e.g., Lowd and Domingos, 2007) very similar to our algorithm for approximate maximum likelihood estimation (Section 6.1), in which we take steps in the direction dictated by the subgradient, scaled by the number of potentials sharing each weight, and return the final average weight vector over

all iterations of learning. EM and primal subgradient solve inference problems to convergence for each update of the parameters, but we warm-start them at each iteration from the optima for the previous iteration to avoid artificially inflating their running times. During learning, the regularization parameter λ is 0.01, and the ADMM parameter ρ is 1.0. These parameters were selected with some light tuning on development sets. The differences among the performances of the learners were not sensitive to changes. For EM, during each M step, we fit the parameters by taking ten subgradient steps, using the MPE state of $P(\mathbf{y}, \mathbf{z}|\mathbf{x}; \mathbf{w})$ to estimate $\mathbb{E}[\phi(\mathbf{y}, \mathbf{z}|\mathbf{x}; w)]$ in the maximum likelihood gradient.

7.4.1 Discovering Latent Groups in Social Media

Groups of people can form online around common traits, interests, or opinions. Often these groups are not explicitly defined in social media, but can be discovered by modeling group membership as latent variables that depend on user behavior. To test paired-dual learning on this task, we collected roughly 4.275M tweets from about 1.350M Twitter users, from a 48-hour window around the Venezuelan presidential election on Oct. 7, 2012. The two major candidates were Hugo Chávez, the incumbent, and Henrique Capriles. We model the supporters of the two candidates by introducing two latent groups.

In order to model membership in these groups, we must construct an SP task for which modeling the groups will improve predictive performance. We construct a model to predict a set of users' interactions with a smaller set of *top users* of interest,

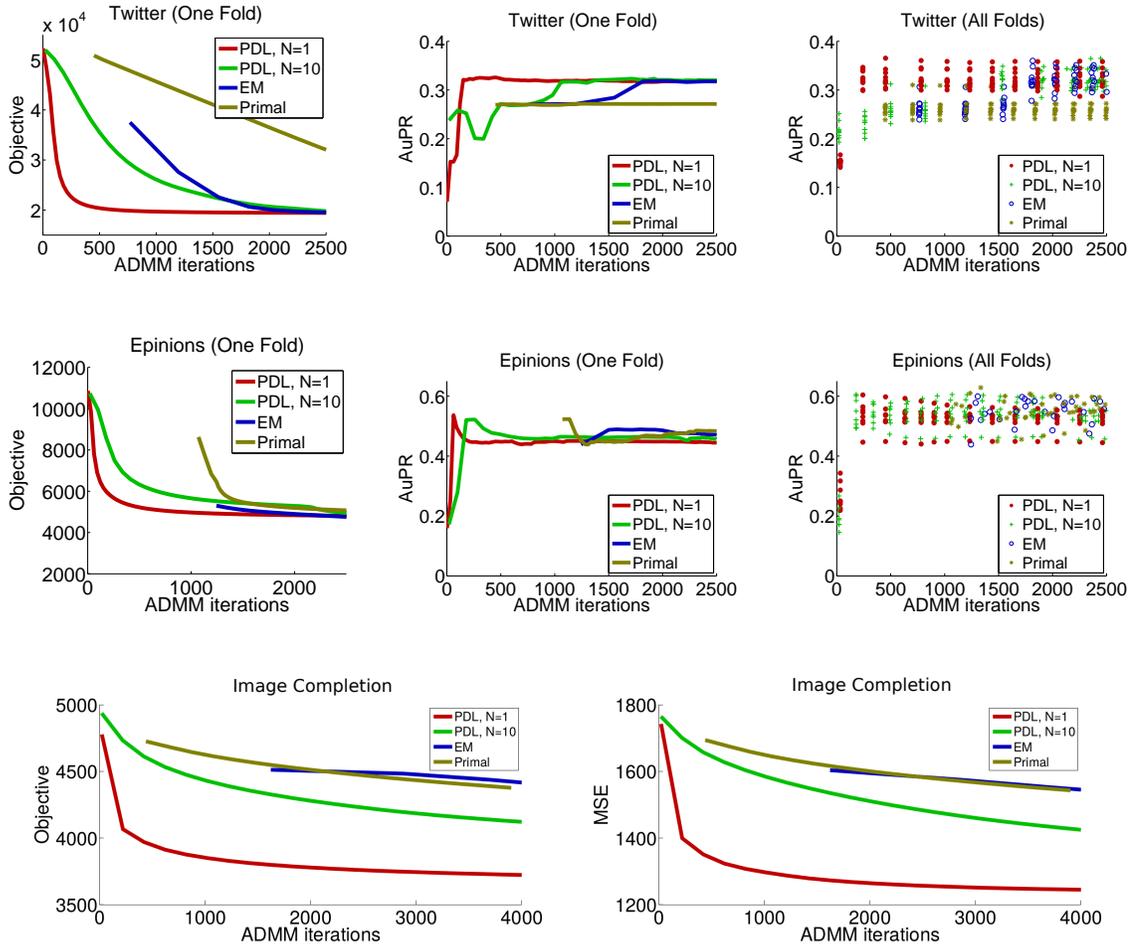


Figure 7.1: Objective score and performance with respect to ADMM iterations for one fold, as well as a subset of points for all folds. On all three problems—group detection, trust prediction, and image completion—paired-dual learning (PDL) reduces the primal learning objective and improves predictive performance much faster than expectation maximization (EM) or primal subgradient (Primal), often reaching a good model before the existing algorithms complete their first parameter update. Full results are in Appendix B.

e.g., political figures, news organizations, and entertainment accounts, given the users' hashtag usage and their interactions with *regular users*, others outside the set of top users. Since our data set focuses on a presidential election, we assume that there are two latent groups, one associated with each major candidate, and we will interpret the learned parameters as the strengths of associations between each group and particular hashtags or top users. We filter the regular users to include only those that used at least one hashtag and interacted with at least one top user in the data, leaving 1,678 users.

Whether each regular user tweeted a hashtag is represented with the PSL predicate `UsedHashtag/2`. Tweets that mention or retweet a top user are not counted. For example, if we observe that User "A" tweeted a tweet that contains the hashtag "#hayuncamino" then `UsedHashtag("A", "#hayuncamino")` has an observed truth value of 1.0. The PSL predicate `RegularUserLink/2` represents whether a regular user retweeted or mentioned *any* user in the full data set that is not a top user, regardless of whether that mentioned or retweeted user is a regular user. Whether a regular user retweeted or mentioned a top user is represented with the PSL predicate `TopUserLink/2`. Finally, the latent group membership of each regular user is represented with the PSL predicate `InGroup/2`.

The dependencies share parameters such that there is a parameter for each hashtag-group pair and each group-top-user interaction pair. We evaluate each model's ability to predict interactions with top users, measuring the area under the precision recall curve (AuPR) using ten folds of cross-validation. In this experiment, we set $K = 0$, immediately starting learning.

7.4.1.1 Latent Group Model

When defining our model, let H be the set of hashtags used by at least 15 different regular users ($|H| = 33$), let T be the set of top users ($|T| = 20$), and let $\mathcal{G} = \{g_0, g_1\}$ be the set of latent groups.

We first include rules that relate hashtag usage to group membership. For each hashtag in H and each latent group, we include a rule of the form

$$w_{h,g} : \text{UsedHashtag}(U, "h") \rightarrow \text{InGroup}(U, "g") \wedge 2$$
$$(\forall "h" \in H, \forall "g" \in \mathcal{G})$$

so that there is a different rule weight governing how strongly each commonly used hashtag is associated with each latent group. Second, we include a rule associating social interactions with group commonality:

$$w_{\text{social}} : \text{RegularUserLink}(U1, U3) \ \&\& \ \text{RegularUserLink}(U2, U3)$$
$$\ \&\& \ U1 \neq U2 \ \&\& \ \text{InGroup}(U1, G) \rightarrow \text{InGroup}(U2, G) \wedge 2$$

This rule encodes the intuition that regular users who interact with the same people on Twitter are more likely to belong to the same latent group. Adding this rule leverages one the advantages of general log-linear models with latent variables: the ability to easily include dependencies among latent variables. Third, we include

rules of the form

$$w_{g,t} : \text{InGroup}(U, "g") \rightarrow \text{TopUserLink}(U, "t") \wedge 2$$

$$(\forall "g" \in \mathcal{G}, \forall "t" \in T)$$

for each latent group and each top user so that there is a parameter governing how strongly each latent group tends to interact with each top user. For entropy surrogates we add the following rules, all with fixed weights of 10.0:

$$10 : !\text{InGroup}(U, "g") \wedge 2 \quad (\forall "g" \in \mathcal{G})$$

$$10 : !\text{TopUserLink}(U1, U2) \wedge 2$$

Last, we constrain the INGROUP atoms for each regular user to sum to 1.0, making INGROUP a mixed-membership assignment:

$$\text{InGroup}(U, +G) = 1.0 \ .$$

We specify initial parameters \mathbf{w} by initializing $w_{h,g}$ to 2.0 for all hashtags and groups, w_{social} to 2.0, and $w_{g,t}$ to 5.0 for all top users and groups, *except* two hashtags and two top users which we assign as seeds. We initially associate the top user "hayuncamino" (Henrique Capriles's campaign account) and the hashtag for Capriles's campaign slogan "#hayuncamino" with Group 0 by initializing the parameters associating them with Group 0 to 10.0 and those associating them with Group

1 to 0.0. We initially associate the top user "chavezcandanga" (Hugo Chávez's account) and the hashtag for Chávez's campaign slogan "#elmundoonchavez" with Group 1 in the same way.

7.4.1.2 Predictive Performance

As Figure 7.1 shows, paired-dual learning optimizes the objective value significantly faster than all other methods, and this faster optimization translates to the faster learning of a more accurate model on test data. In fact, the curves for primal subgradient and EM begin at their first parameter updates, so paired-dual learning reaches a high quality model before the primal methods update their parameters for the first time. Since the methods (except for PDL when $N = 1$ update their parameters at irregular intervals, averaging scores across folds is not possible. (The updates are irregular because inner inference problems reach convergence after different numbers of iterations.) Therefore, the top row of Figure 7.1 plots the objective and AuPR for one fold and a scatter plot of the AuPR on all ten folds for a subset of the points. Results for all folds are in Appendix B.

7.4.1.3 Discovered Groups

In addition to providing strong predictive performance, the learned parameters are interesting for their interpretability. To examine them, we retrain an HL-MRF on the full data set (all ten folds) with paired-dual learning.

Figure 7.2 shows the differences in learned parameters ($w_{h,g_0} - w_{h,g_1}$) associ-

ating hashtag usage with latent groups (excluding the two seeded hashtags). The hashtags are sorted by differences in parameter values from Capriles to Chávez. Our assignment of seeds associated pro-Capriles users with Group 0 and pro-Chávez users with Group 1. The results show a very clean ordering of hashtags based on ideology. Many of the hashtags most strongly associated with the latent Capriles group are explicitly pro-Capriles, e.g., "#mivotoesxcapriles", "#votemosdeprimeroxcapriles", and "#hayuncamimo", an alternative spelling of Capriles's campaign slogan. Others are also clearly anti-Chávez: "#venezueladeluto" ("Venezuela in mourning" after Chávez's reelection) and "#hugochavezfríastequedaldía" (roughly "Hugo Chávez has one day left").

One surprising result is that "#6añosmas" ("six more years") is strongly associated with the latent Capriles group despite superficially appearing to support the incumbent. However, upon inspection of the tweets that use this hashtag, most in our data set use it ironically, predicting "six more years" of "poverty, marginalization, anarchy, crime, corruption, division, hatred, impunity, death" (roughly translated). On the other hand, the hashtags most strongly associated with the Chávez group are all explicitly pro-Chávez. Interestingly, the semantically neutral hashtags promoting voter turnout, such as "#tuvoto", "#vota", and "#vo7a", are inferred to favor the Capriles group. We believe this trend is the result of the social media campaign for increasing voter turnout, which was stronger from the Capriles side.

Figure 7.3 shows the differences in learned parameters ($w_{t,g_0} - w_{t,g_1}$) associating interactions with top users with latent groups (again excluding the two seeded top users). According to the learned model, users in the latent Capriles group

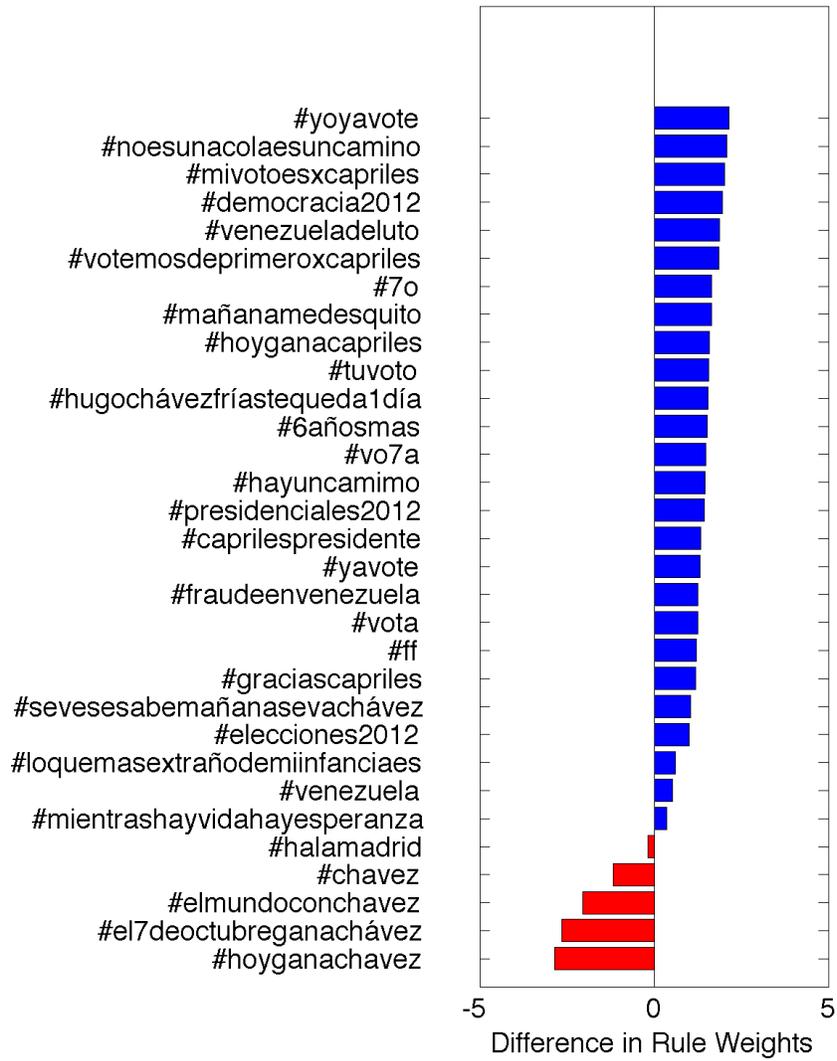


Figure 7.2: Differences in learned parameters associating hashtags with latent groups.

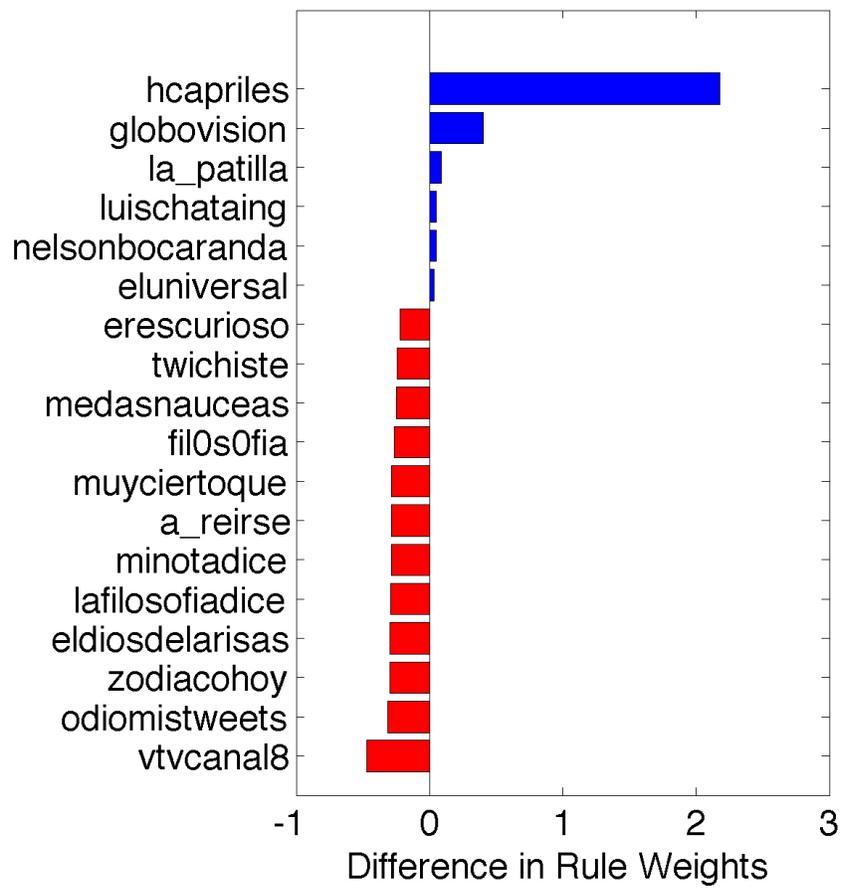


Figure 7.3: Differences in learned parameters associating social interactions with latent groups.

are most likely to interact with "hcapriles" (Capriles's personal account) and independent media outlets and journalists such as "globovision", "la_patilla", "nelsonbocaranda", "luischataing", and "eluniversal". On the other side of the spectrum, users in the latent Chávez group are most likely to interact with "vtvcanal8", the Twitter account of the state-owned television network.

7.4.2 Latent User Attributes in Trust Networks

Next we revisit predicting trust in social networks, which we first considered in Section 6.4.2. In examining the ground-truth data, one finds that distrust links tend not to be evenly distributed throughout the data, but instead are clustered around particular users. This suggests that there might be underlying user attributes that are latent but govern users' behavior as they form trust and distrust links. To model this concept, we introduce two latent attributes for each user, `Trusting/1` and `Trustworthy/1`. We then introduce dependencies between each trusting property and all possible outgoing trust relationships in which the corresponding user participates, and between each trustworthy property and all possible incoming trust relationships. These latent properties act as aggregators, modeling the trends in each user's trust relationships.

As in the fully supervised setting, we use a model that encodes the concept of structural balance in social networks. Instead of only including rules preferring balanced triads, however, we also include rules that prefer unbalance triads, so the learning algorithm can attribute weight to any configuration if it helps optimize its

objective. Removing symmetries, there are 12 distinct logical formulas. Four are for a cyclic structure:

$$w_{cyc}^1 : \text{Trusts}(A,B) \ \&\& \ \text{Trusts}(B,C) \rightarrow \text{Trusts}(C,A) \wedge 2$$

$$w_{cyc}^2 : \text{Trusts}(A,B) \ \&\& \ !\text{Trusts}(B,C) \rightarrow \text{Trusts}(C,A) \wedge 2$$

$$w_{cyc}^3 : \text{Trusts}(A,B) \ \&\& \ \text{Trusts}(B,C) \rightarrow !\text{Trusts}(C,A) \wedge 2$$

$$w_{cyc}^4 : !\text{Trusts}(A,B) \ \&\& \ !\text{Trusts}(B,C) \rightarrow \text{Trusts}(C,A) \wedge 2$$

And eight are for a non-cyclic “v” structure:

$$w_v^1 : \text{Trusts}(A,B) \ \&\& \ \text{Trusts}(B,C) \rightarrow \text{Trusts}(C,B) \wedge 2$$

$$w_v^2 : \text{Trusts}(A,B) \ \&\& \ !\text{Trusts}(B,C) \rightarrow !\text{Trusts}(C,B) \wedge 2$$

$$w_v^3 : !\text{Trusts}(A,B) \ \&\& \ \text{Trusts}(B,C) \rightarrow !\text{Trusts}(C,B) \wedge 2$$

$$w_v^4 : !\text{Trusts}(A,B) \ \&\& \ !\text{Trusts}(B,C) \rightarrow \text{Trusts}(C,B) \wedge 2$$

$$w_v^5 : \text{Trusts}(A,B) \ \&\& \ \text{Trusts}(B,C) \rightarrow !\text{Trusts}(C,B) \wedge 2$$

$$w_v^6 : \text{Trusts}(A,B) \ \&\& \ !\text{Trusts}(B,C) \rightarrow \text{Trusts}(C,B) \wedge 2$$

$$w_v^7 : !\text{Trusts}(A,B) \ \&\& \ \text{Trusts}(B,C) \rightarrow \text{Trusts}(C,B) \wedge 2$$

$$w_v^8 : !\text{Trusts}(A,B) \ \&\& \ !\text{Trusts}(B,C) \rightarrow !\text{Trusts}(C,B) \wedge 2$$

We also include pairwise interactions:

$$w_{\text{pair}}^+ : \text{Trusts}(A,B) \rightarrow \text{Trusts}(B,A) \wedge 2$$

$$w_{\text{pair}}^- : \neg \text{Trusts}(A,B) \rightarrow \neg \text{Trusts}(B,A) \wedge 2$$

To add latent variable reasoning, we add five rules. The rules

$$w_{\text{latent}}^1 : \text{Trusting}(A) \rightarrow \text{Trusts}(A,B) \wedge 2$$

$$w_{\text{latent}}^2 : \text{Trustworthy}(B) \rightarrow \text{Trusts}(A,B) \wedge 2$$

$$w_{\text{latent}}^3 : \text{Trusting}(A) \ \&\& \ \text{Trustworthy}(B) \rightarrow \text{Trusts}(A,B) \wedge 2$$

infer trust from the latent variables, and the rules

$$w_{\text{latent}}^4 : \text{Trusts}(A,B) \rightarrow \text{Trusting}(A) \wedge 2$$

$$w_{\text{latent}}^5 : \text{Trusts}(A,B) \rightarrow \text{Trustworthy}(B) \wedge 2$$

infer the latent values from other trust predictions and observations. All rules are initialized to weights of 1.0. As in the previous evaluation with the Epinions network, the structure of the social network is observed, so these rules are grounded for $\text{Trusts}(A,B)$ atoms where A and B are observed to know each other. In PSL this is implemented by concatenating a $\text{Knows}(A,B)$ atom to the body of each rule with a conjunction, which we omit for readability. For entropy surrogates, we use

the following rules, all with fixed weights of 10.0:

```
10.0 : Trusts(A,B) ^2
10.0 : !Trusts(A,B) ^2
10.0 : Trusting(A,B) ^2
10.0 : !Trusting(A,B) ^2
10.0 : Trustworthy(A,B) ^2
10.0 : !Trustworthy(A,B) ^2
```

We evaluate on the same eight folds of the Epinions data as in Section 6.4.2 and we plot the objective and AuPR curves for held-out distrust relationships from one fold and a scatter plot of the AuPR for a subset of the points for all folds. (We show results for distrust relationships because they account for roughly 10% of all relationships and are therefore harder to predict with high precision and recall.)

The results in Figure 7.1 again show a faster objective descent for paired-dual learning, which learns a high-accuracy model well before the other methods begin learning. Full results are in Appendix B.

7.4.3 Image Completion

Reconstructing part of an obstructed image requires some amount of semantic understanding of physical objects that images depict. These latent semantics make it an ideal test setting for latent variable modeling. In these experiments, we learn

latent variable HL-MRFs that can read half of an image and infer the other half of the image. Again using the 400-image Olivetti face data set, we reveal the top half of each face image to the prediction algorithm, and task it with predicting the bottom half. In Section 6.4, we used fully-observed learning to fit non-latent, or “flat”, HL-MRFs to this task, which were able to reconstruct images with mean-squared error comparable to state-of-the-art methods. These flat models had a large number of parameters for potentials between neighboring pixels and “mirror-image” pixels. Examining the outputs from these HL-MRFs reveals that the models relied heavily on trivial structural patterns, such as face symmetry. This reliance is especially obvious in the completions by flat HL-MRFs for bottom-halves of faces, which seemed to mimic the shadows of mouths by reflecting blurry images of top-half eyes. Latent variables improve performance by learning actual facial structures, rather than exploiting trivial patterns. With all the parameters, variables, and dependencies in the model for each pixel, the efficiency of paired-dual learning becomes critical.

We now use a simpler HL-MRF with a latent layer. We include squared hinge-loss potentials between six latent state variables and the input-half pixel intensities, rounded versions of the input pixels, and, finally, the output-half intensities. These potentials allow the values of the latent variables to mediate interactions between the inputs and outputs. We additionally include potentials between each latent state that prefer contiguous regions of latent states, a prior potential for each pixel to learn an average or background value, and a quadratic prior on all free variables, which serves as a surrogate entropy. We omit any direct dependencies between output pixels to isolate the effectiveness of latent variable modeling.

Our model reasons over variables representing the brightness of pixel values `Bright/2`, a binary, thresholded brightness of observed pixels (i.e., an indicator of whether have intensity greater than 0.5) `Binary/2`, and a set of six latent states `LatState/2`. The intuition behind the model is that the observed pixel intensities and the thresholded intensities provide evidence about which latent states are active for a particular image, and these latent states imply patterns in the output pixels. In the following PSL rules, `I` ranges over the images in the data, and `"S_k"` and `"P_ij"` are constants.

For each latent state `"S_k"` and each pixel `"P_ij"`, whether observed or not, we include the rules

$$w_{\text{bright}}^{++}((i, j), k) : \text{LatState}(I, "S_k") \rightarrow \text{Bright}(I, "P_ij") \wedge 2$$

$$w_{\text{bright}}^{+-}((i, j), k) : \text{LatState}(I, "S_k") \rightarrow \neg \text{Bright}(I, "P_ij") \wedge 2$$

$$w_{\text{bright}}^{-+}((i, j), k) : \neg \text{LatState}(I, "S_k") \rightarrow \text{Bright}(I, "P_ij") \wedge 2$$

$$w_{\text{bright}}^{--}((i, j), k) : \neg \text{LatState}(I, "S_k") \rightarrow \neg \text{Bright}(I, "P_ij") \wedge 2$$

For observed pixels, we encode analogous rules for thresholded pixel intensities to

provide more information to the model:

$$w_{\text{binary}}^{++}((i, j), k) : \text{LatState}(I, "S_k") \rightarrow \text{Binary}(I, "P_ij") \wedge 2$$

$$w_{\text{binary}}^{+-}((i, j), k) : \text{LatState}(I, "S_k") \rightarrow \neg \text{Binary}(I, "P_ij") \wedge 2$$

$$w_{\text{binary}}^{-+}((i, j), k) : \neg \text{LatState}(I, "S_k") \rightarrow \text{Binary}(I, "P_ij") \wedge 2$$

$$w_{\text{binary}}^{--}((i, j), k) : \neg \text{LatState}(I, "S_k") \rightarrow \neg \text{Binary}(I, "P_ij") \wedge 2$$

For every pair of latent states "S_i" and "S_j", we include rules to encode their tendency or aversion to co-occur:

$$w_{\text{state}}^{+}(i, j) : \text{LatState}(I, "S_i") \rightarrow \text{LatState}(I, "S_j") \wedge 2$$

$$w_{\text{state}}^{-}(i, j) : \text{LatState}(I, "S_i") \rightarrow \neg \text{LatState}(I, "S_j") \wedge 2$$

Finally, we use fixed-weight priors on the free variables:

$$1.0 : \text{LatState}(I, S) \wedge 2$$

$$1.0 : \neg \text{LatState}(I, S) \wedge 2$$

$$1.0 : \text{Bright}(I, P) \wedge 2$$

$$1.0 : \neg \text{Bright}(I, P) \wedge 2$$

which serve as surrogate entropies.

We train on 50 randomly selected images from the first 350, and test on the last 50 images as was done previously. Because of the higher dimensionality of these

pixel-based models, we set $K = 10$, allowing the ADMM variables to warm up before updating the parameters \mathbf{w} . (These warmup ADMM iterations are included in the plots above.)

We initialize weights using a heuristic to fit the latent states to individual training images. We first compute the average pixel intensities among all training images, then for each latent state S_k , we randomly choose a seed image. We set the positively correlated binary pixel rule weights w_{binary}^{++} and w_{binary}^{--} to 1.0 if the seed image pixel intensity is higher than the average, and the negatively correlated binary pixel rules w_{binary}^{+-} and w_{binary}^{-+} to 1.0 if the seed image pixel is dimmer than the average. This scheme makes the initial model assign corresponding latent features to images that share bright and dark pixel locations with the seed images. Starting with this initialization, which includes no information about the unthresholded pixel intensities, the learning algorithms fit the models to also predict pixel intensity.

Again, paired-dual learning with one iteration of ADMM is significantly faster at optimizing the objective, which directly translates to a reduction in test error, while the primal methods and the more conservative 10-iteration paired-dual approach are much slower to improve the objective, as seen in Figure 7.1. The learned latent variable model fits latent states to archetypal face shapes, as visualized in Figure 7.4.



(a) Bottom-half model



(b) Example completions

Figure 7.4: Visual representation of learned face models and outputs. In (a), we visualize the six latent states learned by the model. The images plot the quad root (to enhance contrast at low values) of learned weights for the six latent states. The top row depicts the weights of potentials preferring bright pixels and the bottom row depicts the weights of potentials preferring dim pixel intensities. In (b), we compare the completions of bottom-half faces. The left column is the original, and the middle and right are the latent and flat HL-MRF, respectively.

Chapter 8: Conclusion

In this thesis I introduced HL-MRFs, a new class of probabilistic graphical models that unite and generalize several approaches to modeling relational and structured data: Boolean logic, probabilistic graphical models, and fuzzy logic. They can capture relaxed, probabilistic inference with Boolean logic and exact, probabilistic inference with fuzzy logic, making them useful models for both discrete and continuous data. HL-MRFs also generalize these inference techniques with additional expressivity, allowing for even more flexibility. HL-MRFs are a significant addition to the library of machine learning models because they embody a very useful point in the spectrum of models that trade off between scalability and expressivity. As I showed, they can be easily applied to a wide range of structured problems in machine learning and achieve high-quality predictive performance, competitive with or surpassing the performance of state-of-the-art models. However, other models do not scale nearly as well, like discrete MRFs, or are not as versatile in their ability to capture such a wide range of problems, like Bayesian probabilistic matrix factorization.

I also introduced PSL, a probabilistic programming language for HL-MRFs. PSL makes HL-MRFs easy to design, allowing users to encode their ideas for struc-

tural dependencies using an intuitive syntax based on first-order logic. Our empirical evaluation shows how easy this process is for the user. PSL also helps improve a very time-consuming aspect of the modeling process: refining a model. In contrast with other types of models that require specialized inference and learning algorithms depending on which structural dependencies are included, HL-MRFs can encode many types of dependencies and scale very well with the same inference and learning algorithms. PSL makes it easy to quickly add, remove, and modify dependencies in the model and rerun inference and learning, allowing users to quickly improve the quality of their models. Finally, because PSL uses a first-order syntax, each PSL program actually specifies an entire class of HL-MRFs, parameterized by the particular data set over which it is grounded. Therefore, a model or components of a model refined for one data set can easily be applied to others.

Next, I introduced inference and learning algorithms that scale to very large problems. The MAP inference algorithm is far more scalable than standard tools for convex optimization because it leverages the sparsity that is so common to the structural dependencies in SP problems. I also showed that the inference algorithm is a useful alternative to commonly used message passing algorithm for approximate inference in discrete MRFs, because it finds the primal solution to the local consistency relaxation objective for models defined using logical clauses. In addition, I introduced algorithms for supervised learning and learning with latent variables. The supervised learning algorithms extend standard learning objectives to HL-MRFs. Paired-dual learning, a framework for learning with latent variables, overcomes the inference bottleneck associated with this task, making latent vari-

ables easy to include in HL-MRFs and still scale up to big problems. Together, this combination of an expressive formalism, a user-friendly probabilistic programming language, and highly scalable algorithms enables researchers and practitioners to easily build large-scale, accurate models of relational and structured data.

This thesis also lays the foundation for many lines of future work. Our analysis of local consistency relaxation (LCR) as a hierarchical optimization is a general proof technique, and it could be used to derive compact forms for other LCR objectives. As in the case of MRFs defined using logical clauses, such compact forms can simplify analysis and could lead to a greater understanding of LCR for other classes of MRFs. Another important line of work is understanding what guarantees apply to the MAP states of HL-MRFs. Can anything be said about their ability to approximate MAP inference in discrete models that go beyond the models already covered by the known rounding guarantees? Future directions also include developing new algorithms for HL-MRFs. One important direction is marginal inference for HL-MRFs and algorithms for sampling from them. Unlike marginal inference for discrete distributions, which computes the marginal probability that a variable is in a particular state, marginal inference for HL-MRFs requires finding the marginal probability that a variable is in a particular range. One option for doing so, as well as generating samples from HL-MRFs, is to extend the hit-and-run sampling scheme of Broecheler and Getoor (2010). This method was developed for continuous constrained MRFs with piecewise-linear potentials. There are also many new domains to which HL-MRFs and PSL can be applied. By providing these modeling tools to other researchers, I have enabled the design and application of new solutions to SP

problems.

Finally, I note that HL-MRFs and PSL are already having an impact on many problem domains, including automatic knowledge base construction (Pujara et al., 2013), high-level computer vision (London et al., 2013b), drug discovery (Fakhraei et al., 2014), natural language semantics (Beltagy et al., 2014; Sridhar et al., 2015), automobile-traffic modeling (Chen et al., 2014), and user attribute (Li et al., 2014) and trust (Huang et al., 2013; West et al., 2014) prediction in social networks. The ability to easily incorporate latent variables into HL-MRFs and PSL has enabled innovative applications, including modeling latent topics in text (Foulds et al., 2015), and improving student outcomes in massive open online courses (MOOCs) by modeling latent information about students and their communications (Ramesh et al., 2014, 2015). Researchers have also studied how to make HL-MRFs and PSL even more scalable by developing distributed implementations (Miao et al., 2013; Magliacane et al., 2015). That they are already being widely applied underscores the conclusion of this thesis: HL-MRFs and PSL directly address an open need of the machine learning community.

Appendix A: Proof of Theorem 2

In this appendix, we prove the equivalence of objectives (3.7) and (3.10). Our proof analyzes the local consistency relaxation to derive an equivalent, more compact optimization over only the variable pseudomarginals $\boldsymbol{\mu}$ that is identical to the MAX SAT relaxation. Since the variables are Boolean, we refer to each pseudomarginal $\mu_i(1)$ as simply μ_i . Let \boldsymbol{x}_j^F denote the unique setting such that $\phi_j(\boldsymbol{x}_j^F) = 0$. (I.e., \boldsymbol{x}_j^F is the setting in which each literal in the clause C_j is false.)

We begin by reformulating the local consistency relaxation as a hierarchical optimization, first over the variable pseudomarginals $\boldsymbol{\mu}$ and then over the factor pseudomarginals $\boldsymbol{\theta}$. Due to the structure of local polytope \mathbb{L} , the pseudomarginals $\boldsymbol{\mu}$ parameterize inner linear programs that decompose over the structure of the MRF, such that—given fixed $\boldsymbol{\mu}$ —there is an independent linear program $\hat{\phi}_j(\boldsymbol{\mu})$ over $\boldsymbol{\theta}_j$ for each clause C_j . We rewrite objective (3.10) as

$$\arg \max_{\boldsymbol{\mu} \in [0,1]^n} \sum_{C_j \in \mathcal{C}} \hat{\phi}_j(\boldsymbol{\mu}), \tag{A.1}$$

where

$$\hat{\phi}_j(\boldsymbol{\mu}) = \max_{\boldsymbol{\theta}_j} w_j \sum_{\mathbf{x}_j | \mathbf{x}_j \neq \mathbf{x}_j^F} \theta_j(\mathbf{x}_j) \quad (\text{A.2})$$

$$\text{such that } \sum_{\mathbf{x}_j | x_j(i)=1} \theta_j(\mathbf{x}_j) = \mu_i \quad \forall i \in I_j^+ \quad (\text{A.3})$$

$$\sum_{\mathbf{x}_j | x_j(i)=0} \theta_j(\mathbf{x}_j) = 1 - \mu_i \quad \forall i \in I_j^- \quad (\text{A.4})$$

$$\sum_{\mathbf{x}_j} \theta_j(\mathbf{x}_j) = 1 \quad (\text{A.5})$$

$$\theta_j(\mathbf{x}_j) \geq 0 \quad \forall \mathbf{x}_j. \quad (\text{A.6})$$

It is straightforward to verify that objectives (3.10) and (A.1) are equivalent for MRFs with disjunctive clauses for potentials. All constraints defining \mathbb{L} can be derived from the constraint $\boldsymbol{\mu} \in [0, 1]^n$ and the constraints in the definition of $\hat{\phi}_j(\boldsymbol{\mu})$. We have omitted redundant constraints to simplify analysis.

To make this optimization more compact, we replace each inner linear program $\hat{\phi}_j(\boldsymbol{\mu})$ with an expression that gives its optimal value for any setting of $\boldsymbol{\mu}$. Deriving this expression requires reasoning about any maximizer $\boldsymbol{\theta}_j^*$ of $\hat{\phi}_j(\boldsymbol{\mu})$, which is guaranteed to exist because problem (A.2) is bounded and feasible¹ for any parameters $\boldsymbol{\mu} \in [0, 1]^n$ and w_j .

We first derive a sufficient condition for the linear program to not be fully satisfiable, in the sense that it cannot achieve a value of w_j , the maximum value of the weighted potential $w_j \phi_j(\mathbf{x})$. Observe that, by the objective (A.2) and the

¹Setting $\theta_j(\mathbf{x}_j)$ to the probability defined by $\boldsymbol{\mu}$ under the assumption that the elements of \mathbf{x}_j are independent, i.e., the product of the pseudomarginals, is always feasible.

simplex constraint (A.5), showing that $\hat{\phi}_j(\boldsymbol{\mu})$ is not fully satisfiable is equivalent to showing that $\theta_j^*(\mathbf{x}_j^F) > 0$.

Lemma 16 *If*

$$\sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i) < 1 , \quad (\text{A.7})$$

then $\theta_j^*(\mathbf{x}_j^F) > 0$.

Proof By the simplex constraint (A.5),

$$\sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i) < \sum_{\mathbf{x}_j} \theta_j^*(\mathbf{x}_j) . \quad (\text{A.8})$$

Also, by summing all the constraints (A.3) and (A.4),

$$\sum_{\mathbf{x}_j | \mathbf{x}_j \neq \mathbf{x}_j^F} \theta_j^*(\mathbf{x}_j) \leq \sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i) , \quad (\text{A.9})$$

because all the components of $\boldsymbol{\theta}^*$ are nonnegative, and—except for $\theta_j^*(\mathbf{x}_j^F)$ —they all appear at least once in constraints (A.3) and (A.4). These bounds imply

$$\sum_{\mathbf{x}_j | \mathbf{x}_j \neq \mathbf{x}_j^F} \theta_j^*(\mathbf{x}_j) < \sum_{\mathbf{x}_j} \theta_j^*(\mathbf{x}_j) , \quad (\text{A.10})$$

which means $\theta_j^*(\mathbf{x}_j^F) > 0$, completing the proof. ■

We next show that if $\hat{\phi}_j(\boldsymbol{\mu})$ is parameterized such that it is not fully satisfiable, as in Lemma 16, then its optimum always takes a particular value defined by $\boldsymbol{\mu}$.

Lemma 17 *If $w_j > 0$ and $\theta_j^*(\mathbf{x}_j^F) > 0$, then*

$$\sum_{\mathbf{x}_j | \mathbf{x}_j \neq \mathbf{x}_j^F} \theta_j^*(\mathbf{x}_j) = \sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i) . \quad (\text{A.11})$$

Proof We prove the lemma via the Karush-Kuhn-Tucker (KKT) conditions (Karush, 1939; Kuhn and Tucker, 1951). Since problem (A.2) is a maximization of a linear function subject to linear constraints, the KKT conditions are necessary and sufficient for any optimum θ_j^* .

Before writing the relevant KKT conditions, we introduce some necessary notation. For a state \mathbf{x}_j , we need to reason about the variables that disagree with the unsatisfied state \mathbf{x}_j^F . Let

$$d(\mathbf{x}_j) \triangleq \{i \in I_j^+ \cup I_j^- | \mathbf{x}_j(i) \neq \mathbf{x}_j^F(i)\} \quad (\text{A.12})$$

be the set of indices for the variables that do not have the same value in the two states \mathbf{x}_j and \mathbf{x}_j^F .

We now write the relevant KKT conditions for θ_j^* . Let $\boldsymbol{\lambda}, \boldsymbol{\alpha}$ be real-valued vectors where $|\boldsymbol{\lambda}| = |I_j^+| + |I_j^-| + 1$ and $|\boldsymbol{\alpha}| = |\theta_j|$. Let each λ_i correspond to a constraint (A.3) or (A.4) for $i \in I_j^+ \cup I_j^-$, and let λ_Δ correspond to the simplex constraint (A.5). Also, let each $\alpha_{\mathbf{x}_j}$ correspond to a constraint (A.6) for each \mathbf{x}_j .

Then, the following KKT conditions hold:

$$\alpha_{\mathbf{x}_j} \geq 0 \quad \forall \mathbf{x}_j \quad (\text{A.13})$$

$$\alpha_{\mathbf{x}_j} \theta_j^*(\mathbf{x}_j) = 0 \quad \forall \mathbf{x}_j \quad (\text{A.14})$$

$$\lambda_\Delta + \alpha_{\mathbf{x}_j^F} = 0 \quad (\text{A.15})$$

$$w_j + \sum_{i \in d(\mathbf{x}_j)} \lambda_i + \lambda_\Delta + \alpha_{\mathbf{x}_j} = 0 \quad \forall \mathbf{x}_j \neq \mathbf{x}_j^F . \quad (\text{A.16})$$

Since $\theta_j^*(\mathbf{x}_j^F) > 0$, by condition (A.14), $\alpha_{\mathbf{x}_j^F} = 0$. By condition (A.15), then $\lambda_\Delta = 0$. From here we can bound the other elements of $\boldsymbol{\lambda}$. Observe that for every $i \in I_j^+ \cup I_j^-$, there exists a state \mathbf{x}_j such that $d(\mathbf{x}_j) = \{i\}$. Then, it follows from condition (A.16) that there exists \mathbf{x}_j such that, for every $i \in I_j^+ \cup I_j^-$,

$$w_j + \lambda_i + \lambda_\Delta + \alpha_{\mathbf{x}_j} = 0 . \quad (\text{A.17})$$

Since $\alpha_{\mathbf{x}_j} \geq 0$ by condition (A.13) and $\lambda_\Delta = 0$, it follows that $\lambda_i \leq -w_j$. With these bounds, we show that, for any state \mathbf{x}_j , if $|d(\mathbf{x}_j)| \geq 2$, then $\theta_j^*(\mathbf{x}_j) = 0$. Assume that for some state \mathbf{x}_j , $|d(\mathbf{x}_j)| \geq 2$. By condition (A.16) and the derived constraints on $\boldsymbol{\lambda}$,

$$\alpha_{\mathbf{x}_j} \geq (|d(\mathbf{x}_j)| - 1)w_j > 0 . \quad (\text{A.18})$$

With condition (A.14), $\theta_j^*(\mathbf{x}_j) = 0$. Next, observe that for all $i \in I_j^+$ (resp. $i \in I_j^-$) and for any state \mathbf{x}_j , if $d(\mathbf{x}_j) = \{i\}$, then $x_j(i) = 1$ (resp. $x_j(i) = 0$), and for any other state \mathbf{x}'_j such that $x'_j(i) = 1$ (resp. $x'_j(i) = 0$), $d(\mathbf{x}'_j) \geq 2$. By constraint (A.3)

(resp. constraint (A.4)), $\theta^*(\mathbf{x}_j) = \mu_i$ (resp. $\theta^*(\mathbf{x}_j) = 1 - \mu_i$).

We have shown that if $\theta_j^*(\mathbf{x}_j^F) > 0$, then for all states \mathbf{x}_j , if $d(\mathbf{x}_j) = \{i\}$ and $i \in I_j^+$ (resp. $i \in I_j^-$), then $\theta_j^*(\mathbf{x}_j) = \mu_i$ (resp. $\theta_j^*(\mathbf{x}_j) = 1 - \mu_i$), and if $|d(\mathbf{x}_j)| \geq 2$, then $\theta_j^*(\mathbf{x}_j) = 0$. This completes the proof. \blacksquare

Lemma 16 says if $\sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i) < 1$, then $\hat{\phi}_j(\boldsymbol{\mu})$ is not fully satisfiable, and Lemma 17 provides its optimal value. We now reason about the other case, when $\sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i) \geq 1$, and we show that it is sufficient to ensure that $\hat{\phi}_j(\boldsymbol{\mu})$ is fully satisfiable.

Lemma 18 *If $w_j > 0$ and*

$$\sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i) \geq 1, \quad (\text{A.19})$$

then $\theta_j^(\mathbf{x}_j^F) = 0$.*

Proof We prove the lemma by contradiction. Assume that $w_j > 0$, $\sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i) \geq 1$, and that the lemma is false, $\theta_j^*(\mathbf{x}_j^F) > 0$. Then, by Lemma 17,

$$\sum_{\mathbf{x}_j | \mathbf{x}_j \neq \mathbf{x}_j^F} \theta_j^*(\mathbf{x}_j) \geq 1. \quad (\text{A.20})$$

The assumption that $\theta_j^*(\mathbf{x}_j^F) > 0$ implies

$$\sum_{\mathbf{x}_j} \theta_j^*(\mathbf{x}_j) > 1, \quad (\text{A.21})$$

which is a contradiction, since it violates the simplex constraint (A.5). The possibility that $\theta_j^*(\mathbf{x}_j^F) < 0$ is excluded by the nonnegativity constraints (A.6). ■

For completeness and later convenience, we also state the value of $\hat{\phi}_j(\boldsymbol{\mu})$ when it is fully satisfiable.

Lemma 19 *If $\theta_j^*(\mathbf{x}_j^F) = 0$, then*

$$\sum_{\mathbf{x}_j | \mathbf{x}_j \neq \mathbf{x}_j^F} \theta_j^*(\mathbf{x}_j) = 1 . \quad (\text{A.22})$$

Proof The lemma follows from the simplex constraint (A.5). ■

We can now combine the previous lemmas into a single expression for the value of $\hat{\phi}_j(\boldsymbol{\mu})$.

Lemma 20 *For any feasible setting of $\boldsymbol{\mu}$,*

$$\hat{\phi}_j(\boldsymbol{\mu}) = w_j \min \left\{ \sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i), 1 \right\} . \quad (\text{A.23})$$

Proof The lemma is trivially true if $w_j = 0$ since any assignment will yield zero value. If $w_j > 0$, then we consider two cases. In the first case, if $\sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i) < 1$, then, by Lemmas 16 and 17,

$$\hat{\phi}_j(\boldsymbol{\mu}) = w_j \left(\sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i) \right) . \quad (\text{A.24})$$

In the second case, if $\sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i) \geq 1$, then, by Lemmas 18 and 19,

$$\hat{\phi}_j(\boldsymbol{\mu}) = w_j . \tag{A.25}$$

By factoring out w_j , we can rewrite this piecewise definition of $\hat{\phi}_j(\boldsymbol{\mu})$ as w_j multiplied by the minimum of $\sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i)$ and 1, completing the proof. ■

This leads to our final equivalence result.

Theorem 2 *For an MRF with potentials corresponding to disjunctive logical clauses and associated nonnegative weights, the first-order local consistency relaxation of MAP inference is equivalent to the MAX SAT relaxation of Goemans and Williamson (1994). Specifically, any partial optimum $\boldsymbol{\mu}^*$ of objective (3.10) is an optimum $\hat{\boldsymbol{y}}^*$ of objective (3.7), and vice versa.*

Proof Substituting the solution of the inner optimization from Lemma 20 into the local consistency relaxation objective (A.1) gives a projected optimization over only $\boldsymbol{\mu}$ which is identical to the MAX SAT relaxation objective (3.7). ■

Appendix B: Additional Results on Learning with Latent Variables

In our experiments with paired-dual learning (PDL) in Chapter 7, we compared different learning methods and measured the value of the objective function and the predictive performance of the learned model after different numbers of inference steps. Averaging these values across folds of the data is not possible because the methods (except for PDL when $N = 1$) update the parameters at irregular intervals. The updates are irregular because inner inference problems reach convergence after different numbers of iterations. We therefore present the full results for all folds for predicting interactions in social media to evaluate our discovered latent groups (Section 7.4.1) and predicting trust in social networks using latent user attributes (Section 7.4.2). Note that there are no additional results for image completion using latent variables (Section 7.4.3) because we performed that evaluation using a standard test set.

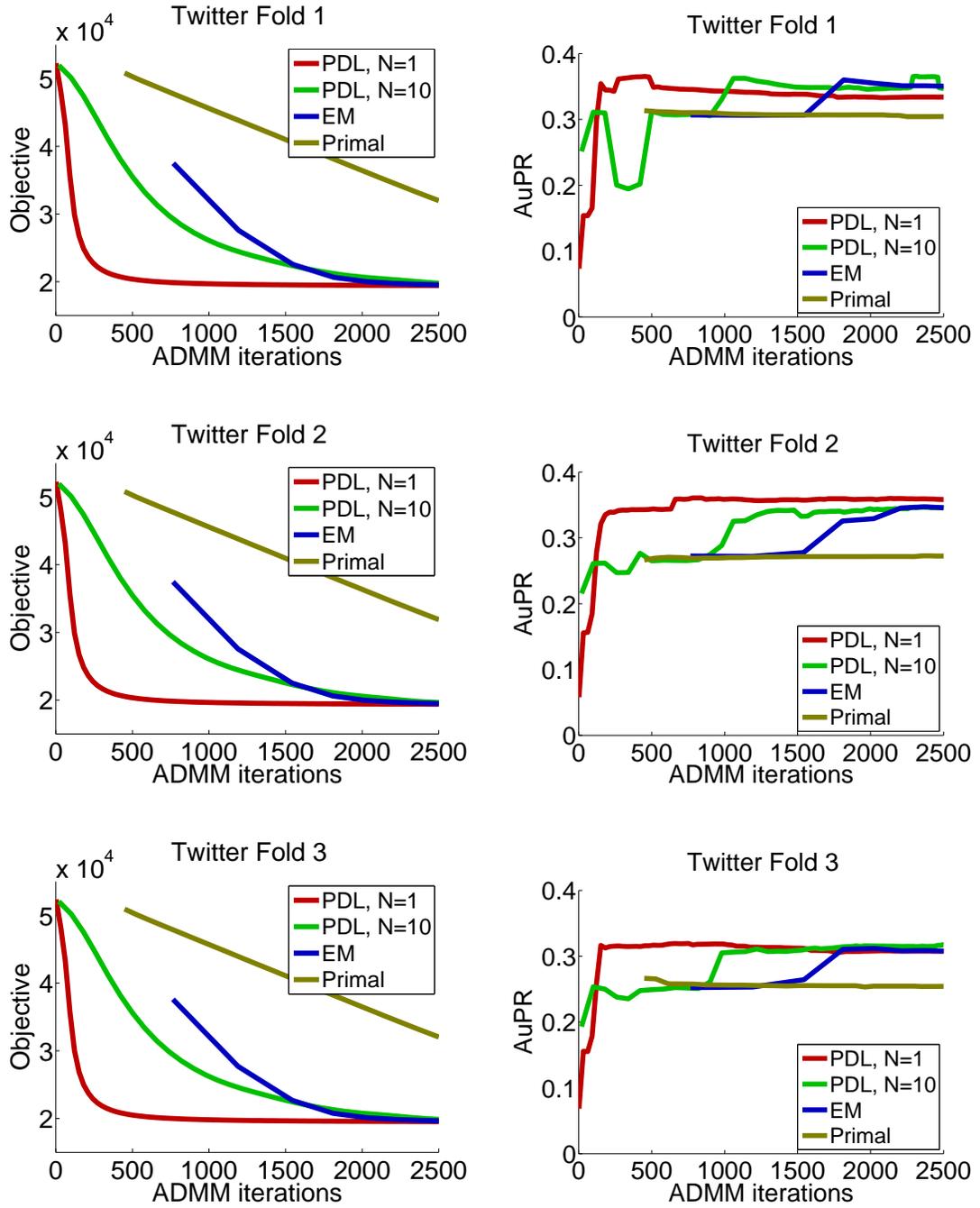


Figure B.1: Results for interaction prediction on Twitter data set, folds 1-3.

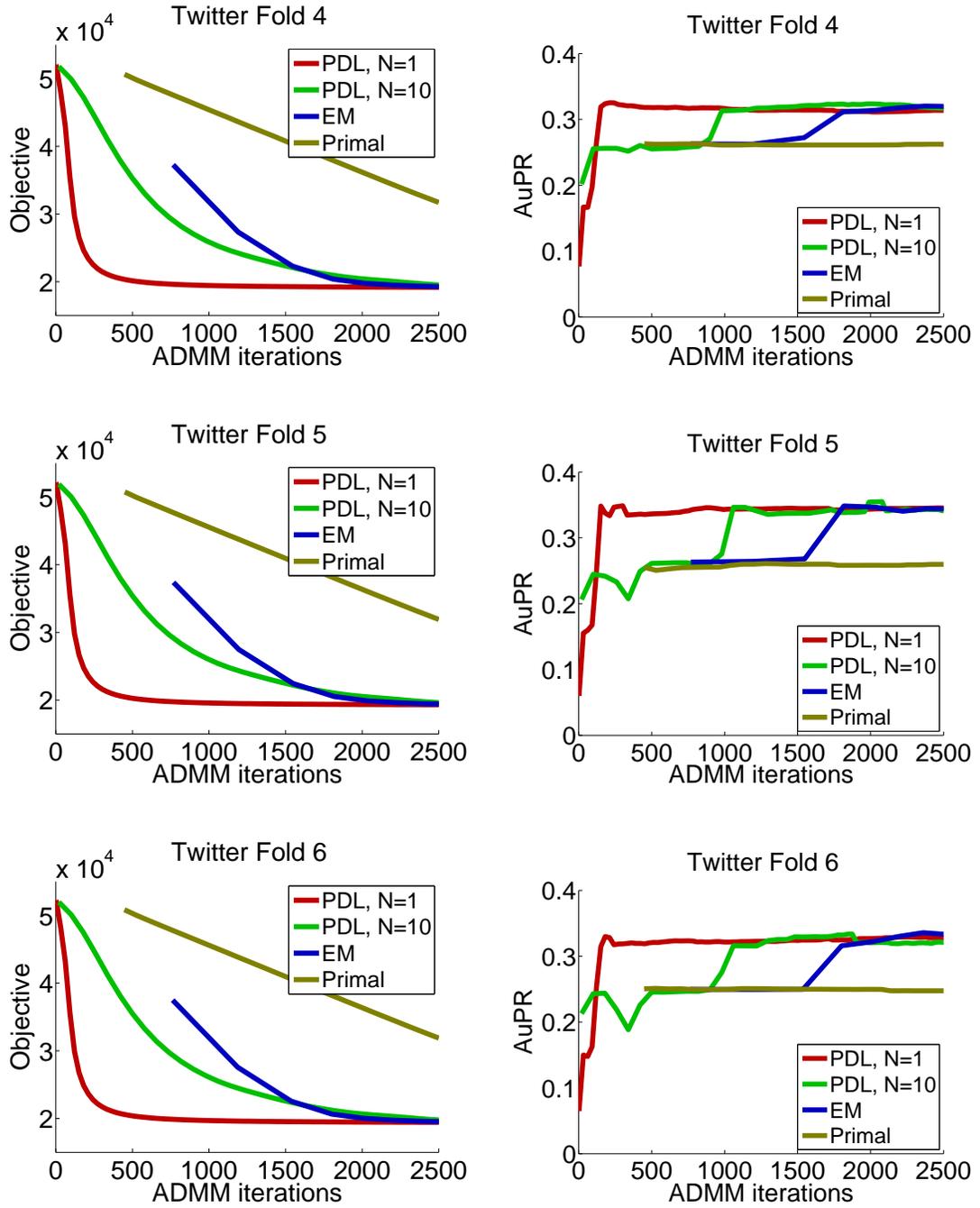


Figure B.2: Results for interaction prediction on Twitter data set, folds 4-6.

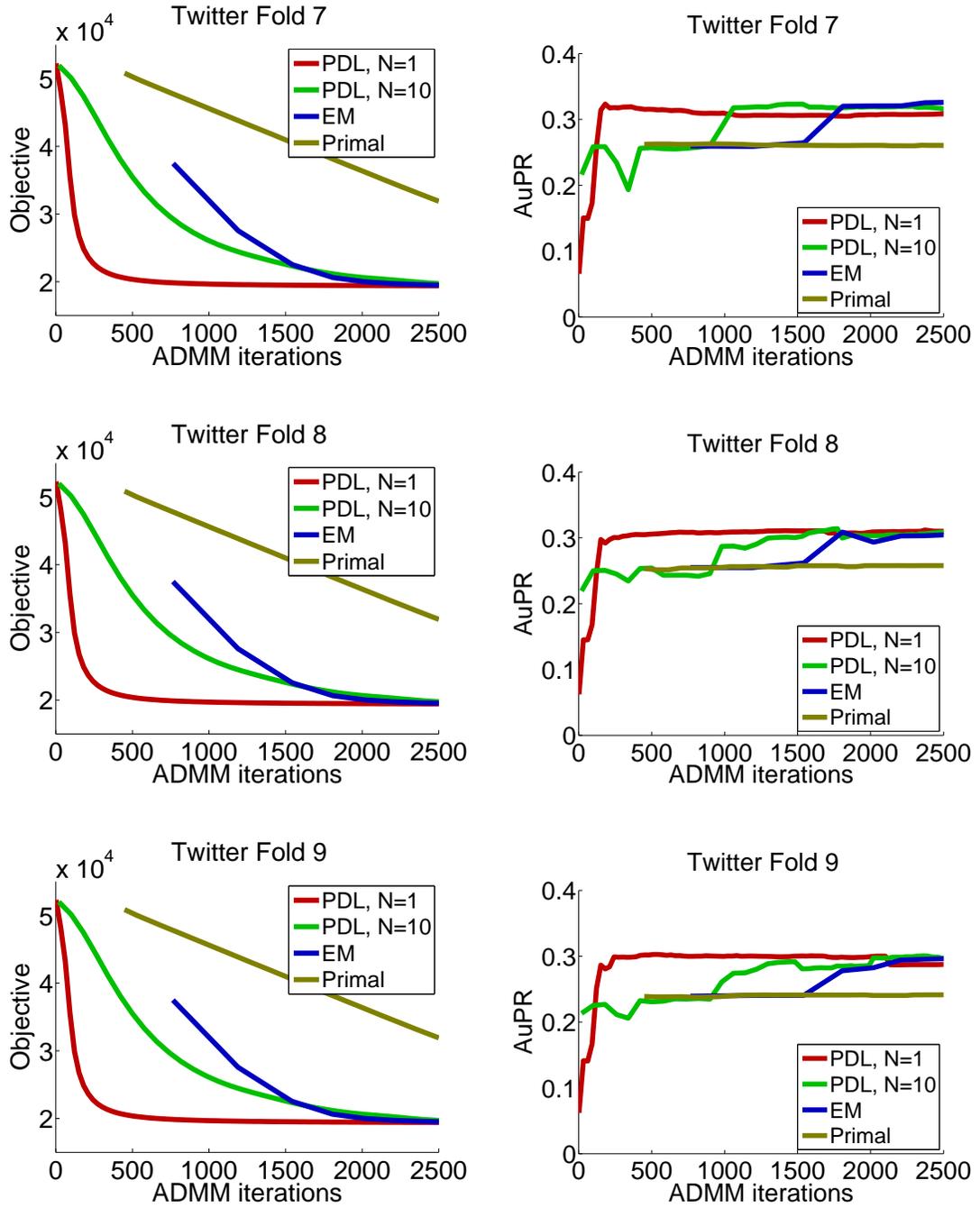


Figure B.3: Results for interaction prediction on Twitter data set, folds 7-9.

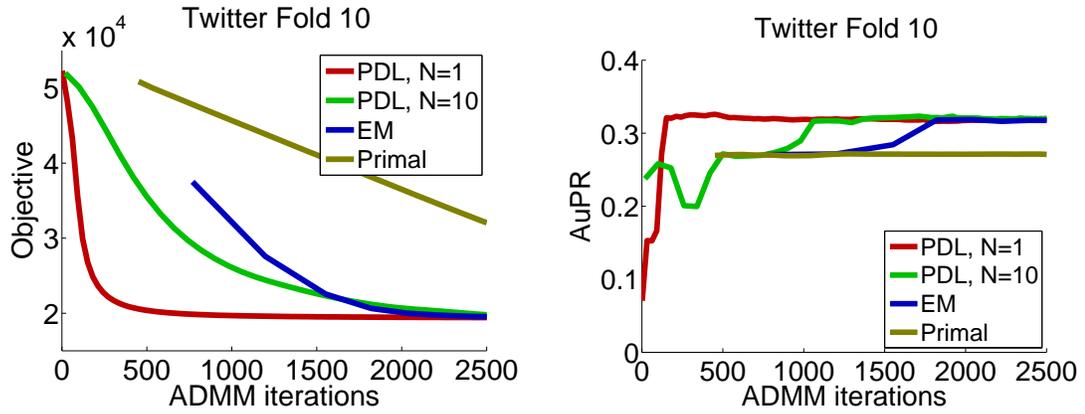


Figure B.4: Results for interaction prediction on Twitter data set, fold 10.

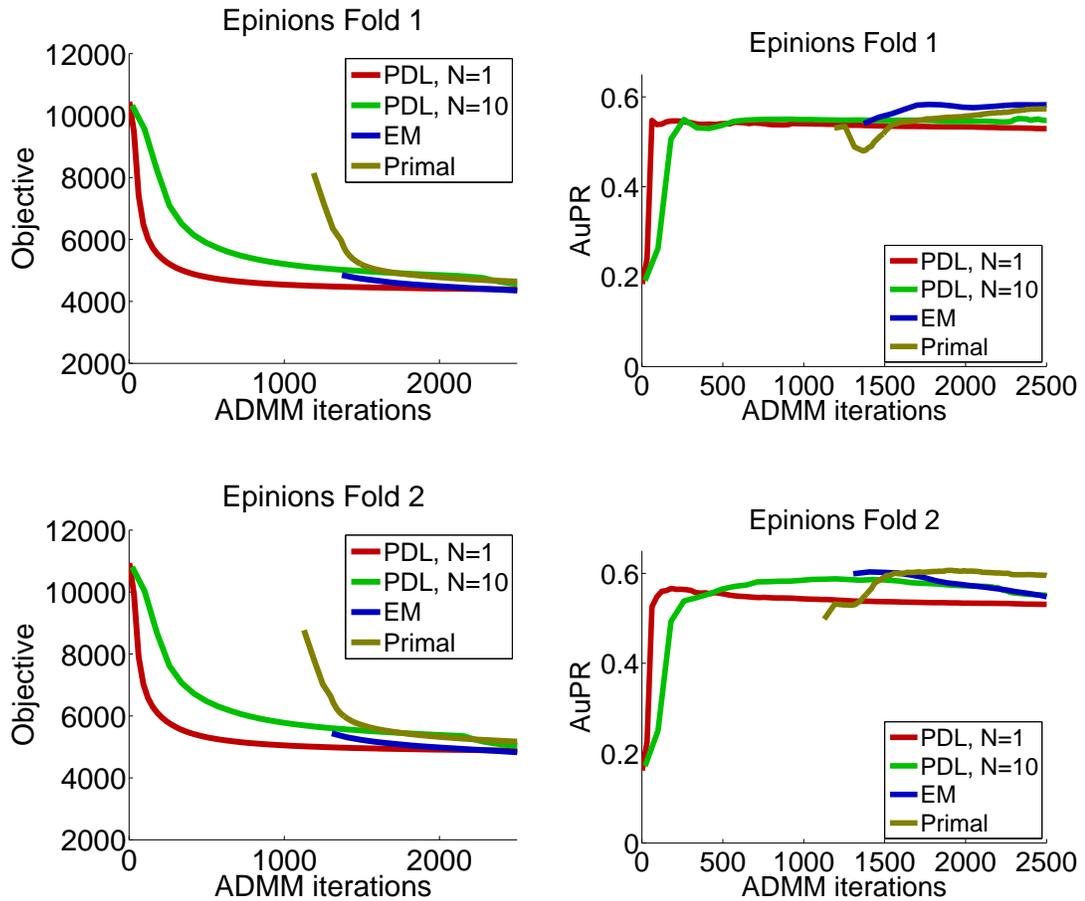


Figure B.5: Results for social-trust prediction on Epinions data set, folds 1-2.

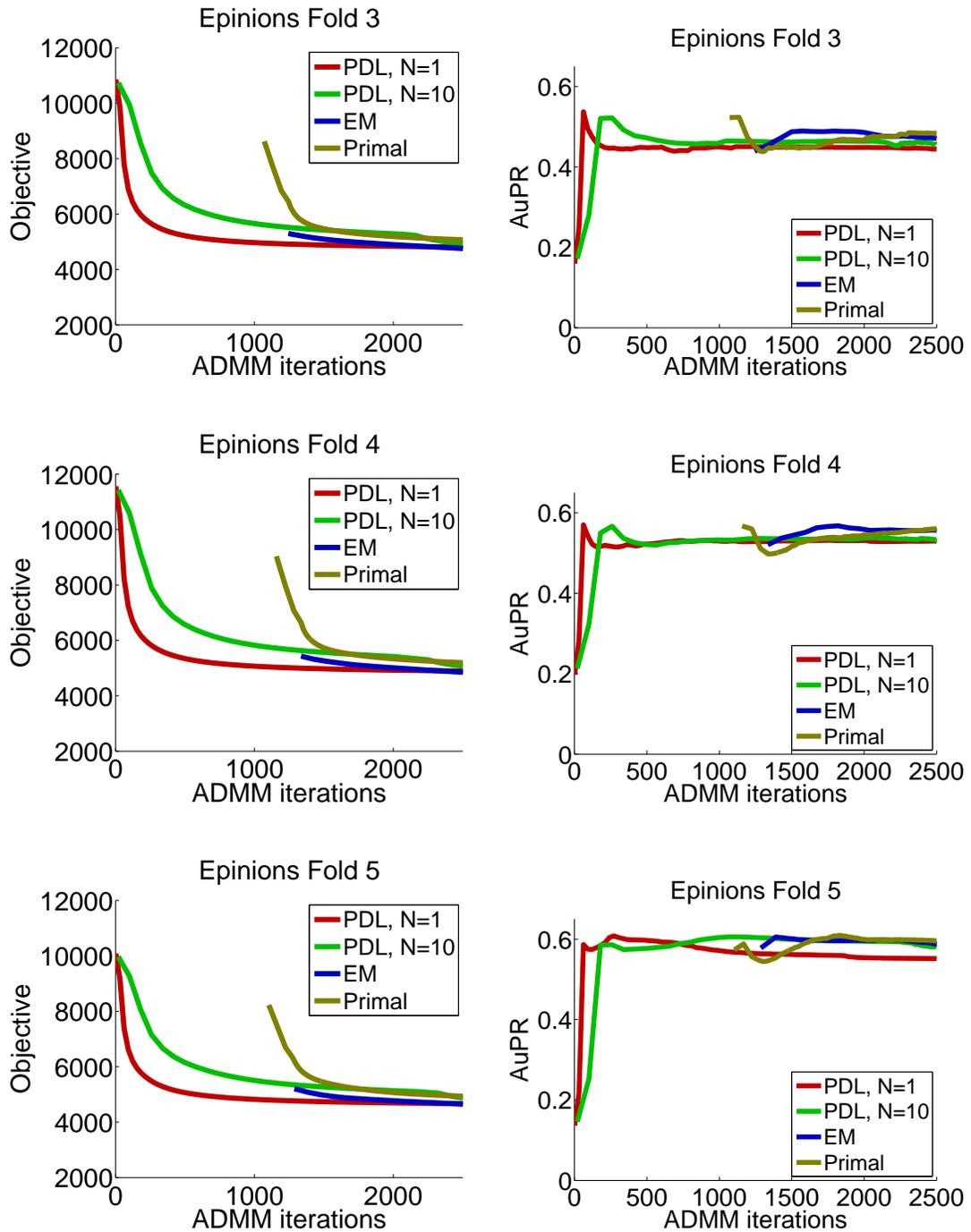


Figure B.6: Results for social-trust prediction on Epinions data set, folds 3-5.

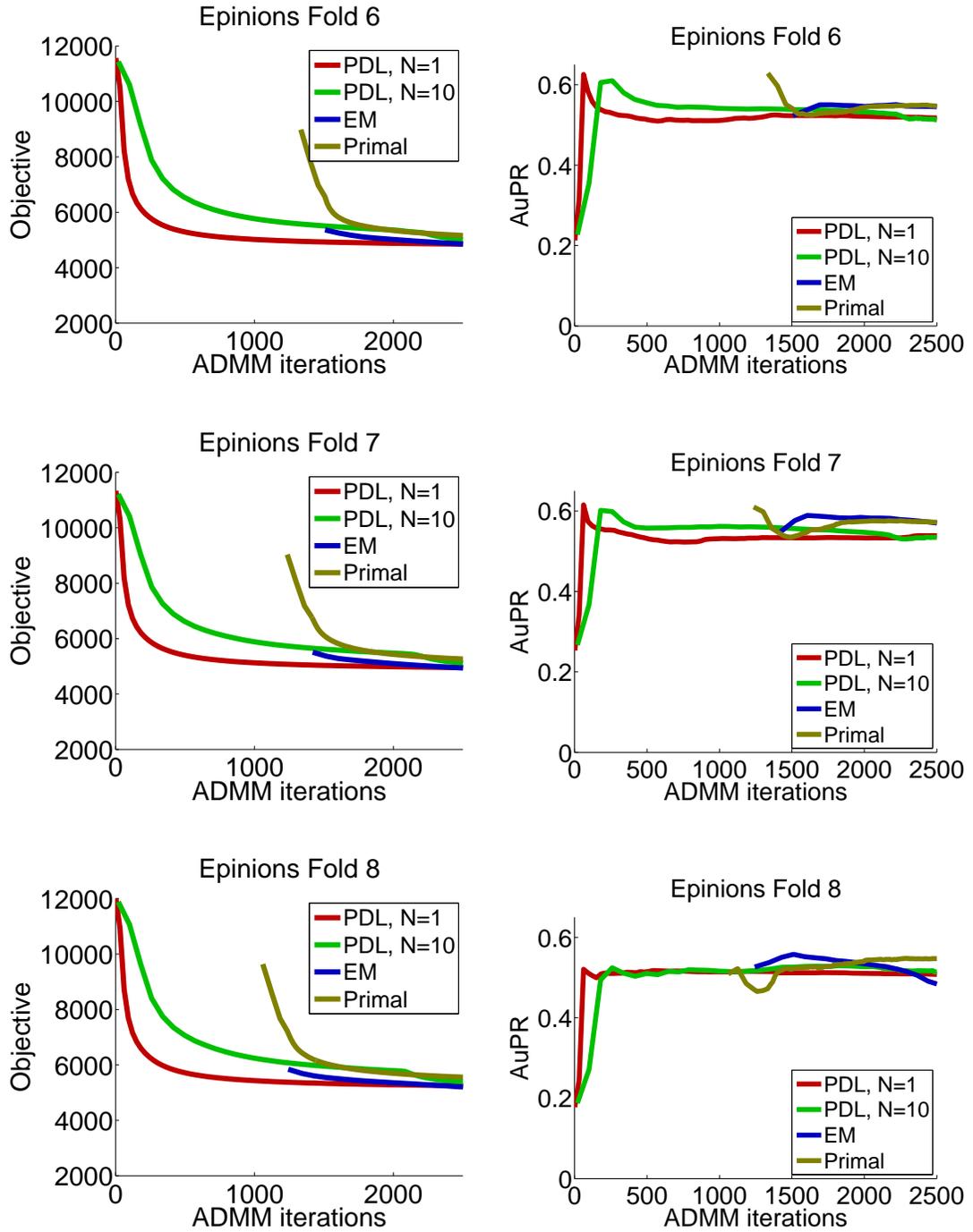


Figure B.7: Results for social-trust prediction on Epinions data set, folds 6-8.

Bibliography

- A. Abdelbar and S. Hedetniemi. Approximating MAPs for belief networks is NP-hard and other theorems. *Artificial Intelligence*, 102(1):21–38, 1998.
- N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley-Interscience, third edition, 2008.
- L. An and P. Tao. The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Annals of Operations Research*, 133:23–46, 2005.
- T. Asano and D. P. Williamson. Improved approximation algorithms for MAX SAT. *J. Algorithms*, 42(1):173–202, 2002.
- S. H. Bach, M. Broecheler, L. Getoor, and D. P. O’Leary. Scaling MPE inference for constrained continuous Markov random fields. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2663–2671, 2012.
- S. H. Bach, B. Huang, B. London, and L. Getoor. Hinge-loss Markov random fields: Convex inference for structured prediction. In *Uncertainty in Artificial Intelligence (UAI)*, 2013.
- S. H. Bach, B. Huang, J. Boyd-Graber, and L. Getoor. Paired-dual learning for fast training of latent variable hinge-loss MRFs. In *International Conference on Machine Learning (ICML)*, 2015a.
- S. H. Bach, B. Huang, and L. Getoor. Unifying local consistency and MAX SAT relaxations for scalable inference with rounding guarantees. In *Artificial Intelligence and Statistics (AISTATS)*, 2015b.
- G. BakIr, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan, editors. *Predicting Structured Data*. MIT Press, 2007.
- I. Beltagy, K. Erk, and R. J. Mooney. Probabilistic soft logic for semantic textual similarity. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2014.

- J. Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society*, 24(3):179–195, 1975.
- S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. *Distributed Optimization and Statistical Learning Via the Alternating Direction Method of Multipliers*. Now Publishers, 2011.
- M. Broecheler and L. Getoor. Computing marginal distributions over continuous Markov networks for statistical relational learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.
- M. Broecheler, L. Mihalkova, and L. Getoor. Probabilistic similarity logic. In *Uncertainty in Artificial Intelligence (UAI)*, 2010a.
- M. Broecheler, P. Shakarian, and V. S. Subrahmanian. A scalable framework for modeling competitive diffusion in social networks. In *Social Computing (Social-Com)*, 2010b.
- C. Chekuri, S. Khanna, J. Naor, and L. Zosin. A linear programming formulation and approximation algorithms for the metric labeling problem. *SIAM J. Discrete Math.*, 18(3):608–625, 2005.
- P. Chen, F. Chen, and Z. Qian. Road traffic congestion monitoring in social media with hinge-loss Markov random fields. In *IEEE International Conference on Data Mining (ICDM)*, 2014.
- A. Choi, T. Standley, and A. Darwiche. Approximating weighted Max-SAT problems by compensating for relaxations. In *International Conference on Principles and Practice of Constraint Programming*, 2009.
- M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2002.
- M. Collins and B. Roark. Incremental parsing with the perceptron algorithm. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2004.
- H. Daumé III, J. Langford, and D. Marcu. Search-based structured prediction. *Machine Learning*, 75(3):297–325, 2009.
- J. Davies and F. Bacchus. Exploiting the power of MIP solvers in MAXSAT. In M. Järvisalo and A. Van Gelder, editors, *Theory and Applications of Satisfiability Testing SAT 2013*, Lecture Notes in Computer Science, pages 166–181. Springer Berlin Heidelberg, 2013.
- L. De Raedt, A. Kimmig, and H. Toivonen. ProbLog: A probabilistic Prolog and its application in link discovery. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

- J. Domke. Structured learning via logistic regression. In *Neural Information Processing Systems (NIPS)*, 2013.
- J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research (JMLR)*, 2011.
- S. Fakhraei, B. Huang, L. Raschid, and L. Getoor. Network-based drug-target interaction prediction with probabilistic soft logic. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2014.
- J. Feldman, M. J. Wainwright, and D. R. Karger. Using linear programming to decode binary linear codes. *Information Theory, IEEE Trans. on*, 51(3):954–972, 2005.
- J. Foulds, N. Navaroli, P. Smyth, and A. Ihler. Revisiting MAP estimation, message passing and perfect graphs. In *AI & Statistics*, 2011.
- J. Foulds, S. Kumar, and L. Getoor. Latent topic networks: A versatile probabilistic programming framework for topic models. In *International Conference on Machine Learning (ICML)*, 2015.
- N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1999.
- D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.
- M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.
- L. Getoor and B. Taskar, editors. *Introduction to statistical relational learning*. MIT press, 2007.
- A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.
- A. Globerson, D. Sontag, D. K. Choe, and Y. Li. MPLP, 2012. URL http://cs.nyu.edu/~dsontag/code/mplp_ver2.tgz.
- R. Glowinski and A. Marrocco. Sur l’approximation, par éléments finis d’ordre un, et la résolution, par pénalisation-dualité, d’une classe de problèmes de Dirichlet non linéaires. *Revue française d’automatique, informatique, recherche opérationnelle*, 9(2):41–76, 1975.
- M. X. Goemans and D. P. Williamson. New $3/4$ -approximation algorithms for the maximum satisfiability problem. *SIAM J. Discrete Math.*, 7(4):656–666, 1994.

- K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.
- N. D. Goodman, V. K. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum. Church: A language for generative models. In *Uncertainty in Artificial Intelligence (UAI)*, 2008.
- A. D. Gordon, T. A. Henzinger, A. V. Nori, and S. K. Rajamani. Probabilistic programming. In *International Conference on Software Engineering (ICSE, FOSE track)*, 2014.
- T. Heskes. Convexity arguments for efficient minimization of the Bethe and Kikuchi free energies. *Journal of Artificial Intelligence Research*, 26:153–190, 2006.
- G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
- B. Huang, A. Kimmig, L. Getoor, and J. Golbeck. A flexible framework for probabilistic models of social trust. In *Conference on Social Computing, Behavioral-Cultural Modeling, & Prediction*, 2013.
- T. Huynh and R. Mooney. Max-margin weight learning for Markov logic networks. In *European Conference on Machine Learning (ECML)*, 2009.
- T. Jebara. MAP estimation, message passing, and perfect graphs. In *Uncertainty in Artificial Intelligence (UAI)*, 2009.
- T. Joachims, T. Finley, and C. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.
- V. Jovic, S. Gould, and D. Koller. Accelerated dual decomposition for MAP inference. In *International Conference on Machine Learning (ICML)*, 2010.
- W. Karush. Minima of Functions of Several Variables with Inequalities as Side Constraints. Master’s thesis, University of Chicago, 1939.
- J. Kleinberg and É. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. *J. ACM*, 49(5):616–639, 2002.
- G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall, 1995.
- S. Kok and P. Domingos. Statistical predicate invention. In *International Conference on Machine Learning (ICML)*, 2007.
- D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

- V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 28(10):1568–1583, 2006.
- N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 33(3):531–552, 2011.
- H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Berkeley Symp. on Math. Statist. and Prob.*, 1951.
- M. P. Kumar, P. H. S. Torr, and A. Zisserman. Solving Markov random fields using second order cone programming relaxations. In *Computer Vision and Pattern Recognition (CVPR)*, 2006.
- N. Landwehr, A. Passerini, L. De Raedt, and P. Frasconi. Fast learning of relational kernels. *Machine Learning*, 78(3):305–342, 2010.
- J. Larrosa, F. Heras, and S. de Givry. A logical approach to efficient Max-SAT solving. *Artificial Intelligence*, 172(2-3):204–233, 2008.
- J. Li, A. Ritter, and D. Jurafsky. Inferring user preferences by probabilistic logical reasoning over social networks. *arXiv preprint arXiv:1411.2679*, 2014.
- B. London, B. Huang, B. Taskar, and L. Getoor. Collective stability in structured prediction: Generalization from one example. In *International Conference on Machine Learning (ICML)*, 2013a.
- B. London, S. Khamis, S. H. Bach, B. Huang, L. Getoor, and L. Davis. Collective activity detection using hinge-loss Markov random fields. In *CVPR Workshop on Structured Prediction: Tractability, Learning and Inference*, 2013b.
- D. Lowd and P. Domingos. Efficient weight learning for Markov logic networks. In *Principles and Practice of Knowledge Discovery in Databases (PKDD)*, 2007.
- S. Magliacane, P. Stutz, P. Groth, and A. Bernstein. FoxPSL: An extended and scalable PSL implementation. In *AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches*, 2015.
- A. F. T. Martins, M. A. T. Figueiredo, P. M. Q. Aguiar, N. A. Smith, and E. P. Xing. AD³: Alternating Directions Dual Decomposition for MAP Inference in Graphical Models. *Journal of Machine Learning Research (JMLR)*, 16(Mar):495–545, 2015.
- A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2000.
- A. McCallum, K. Schultz, and S. Singh. FACTORIE: Probabilistic programming via imperatively defined factor graphs. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

- O. Meshi and A. Globerson. An alternating direction method for dual MAP LP relaxation. In *European Conference on Machine learning (ECML)*, 2011.
- O. Meshi, A. Jaimovich, A. Globerson, and N. Friedman. Convexifying the Bethe free energy. In *Uncertainty in Artificial Intelligence (UAI)*, 2009.
- O. Meshi, D. Sontag, T. Jaakkola, and A. Globerson. Learning efficiently with approximate inference via dual losses. In *International Conference on Machine Learning (ICML)*, 2010.
- E. Mezuman, D. Tarlow, A. Globerson, and Y. Weiss. Tighter linear program relaxations for high order graphical models. In *Uncertainty in Artificial Intelligence (UAI)*, 2013.
- H. Miao, X. Liu, B. Huang, and L. Getoor. A hypergraph-partitioned vertex programming approach for large-scale consensus optimization. In *IEEE International Conference on Big Data*, 2013.
- B. Milch, B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov. BLOG: Probabilistic models with unknown objects. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.
- P. Mills and E. Tsang. Guided local search for solving SAT and weighted MAX-SAT problems. *J. Automated Reasoning*, 24(1-2):205–223, 2000.
- S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.
- R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. Jordan, editor, *Learning in Graphical Models*, pages 355–368. MIT Press, 1999.
- Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.
- J. Neville and D. Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692, 2007.
- H. B. Newcombe and J. M. Kennedy. Record linkage: Making maximum use of the discriminating power of identifying information. *Communications of the ACM*, 5(11):563–566, 1962.
- J. D. Park. Using weighted MAX-SAT engines to solve MPE. In *AAAI Conference on Artificial Intelligence*, 2002.
- A. Pfeffer. IBAL: A probabilistic rational programming language. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.
- A. Pfeffer. Figaro: An object-oriented probabilistic programming language. Technical report, Charles River Analytics, 2009.

- H. Poon and P. Domingos. Unsupervised semantic parsing. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2009.
- H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *Uncertainty in Artificial Intelligence (UAI)*, 2011.
- J. Pujara, H. Miao, L. Getoor, and W. Cohen. Knowledge graph identification. In *International Semantic Web Conference (ISWC)*, 2013.
- A. Ramesh, D. Goldwasser, B. Huang, H. Daumé III, and L. Getoor. Learning latent engagement patterns of students in online courses. In *AAAI Conference on Artificial Intelligence*, 2014.
- A. Ramesh, S. Kumar, J. Foulds, and L. Getoor. Weakly supervised models of aspect-sentiment for online course discussion forums. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015.
- P. Ravikumar and J. Lafferty. Quadratic programming relaxations for metric labeling and Markov random field MAP estimation. In *International Conference on Machine Learning (ICML)*, 2006.
- P. Ravikumar, A. Agarwal, and M. J. Wainwright. Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *J. Mach. Learn. Res.*, 11:1043–1080, 2010.
- M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.
- M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In D. Fensel, K. Sycara, and J. Mylopoulos, editors, *The Semantic Web - ISWC 2003*, volume 2870 of *Lecture Notes in Computer Science*, pages 351–368. Springer Berlin / Heidelberg, 2003.
- S. Ross and J. A. Bagnell. Reinforcement and Imitation Learning via Interactive No-Regret Learning, 2014.
- S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Artificial Intelligence & Statistics (AISTATS)*, 2011.
- R. Salakhutdinov and G. Hinton. Deep Boltzmann machines. In *Artificial Intelligence & Statistics (AISTATS)*, 2009.
- R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *International Conference on Machine Learning (ICML)*, 2008.
- A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer-Verlag, 2003.

- A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Efficient structured prediction with latent variables for general graphical models. In *International Conference on Machine Learning (ICML)*, 2012a.
- A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Globally convergent dual MAP LP relaxation solvers using Fenchel-Young margins. In *Advances in Neural Information Processing Systems (NIPS)*, 2012b.
- P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.
- S. E. Shimony. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68(2):399–410, 1994.
- D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *Uncertainty in Artificial Intelligence (UAI)*, 2008.
- D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*, pages 219–254. MIT Press, 2011.
- D. Sontag, D. K. Choe, and Y. Li. Efficiently searching for frustrated cycles in MAP inference. In *Uncertainty in Artificial Intelligence (UAI)*, 2012.
- D. Sridhar, J. Foulds, M. Walker, B. Huang, and L. Getoor. Joint models of disagreement and stance in online debate. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015.
- B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Neural Information Processing Systems (NIPS)*, 2004.
- B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *International Conference on Machine Learning (ICML)*, 2005.
- I. Tsochantaris, T. Joachims, T. Hofmann, and Y. Altun. Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, 2005.
- L. van der Maaten, M. Welling, and L. Saul. Hidden-unit conditional random fields. In *Artificial Intelligence & Statistics*, 2011.
- V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 2000.
- B. W. Wah and Y. Shang. Discrete Lagrangian-based search for solving MAX-SAT problems. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1997.

- M. J. Wainwright and M. I. Jordan. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers, 2008.
- Y. Weiss, C. Yanover, and T. Meltzer. MAP estimation, linear programming and belief propagation with convex free energies. In *Uncertainty in Artificial Intelligence (UAI)*, 2007.
- T. Werner. A linear programming approach to max-sum problem: A review. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 29(7):1165–1179, 2007.
- R. West, H. S. Paskov, J. Leskovec, and C. Potts. Exploiting social network structure for person-to-person sentiment analysis. *Transactions of the Association for Computational Linguistics (TACL)*, 2:297–310, 2014.
- M. Wright. The interior-point revolution in optimization: History, recent developments, and lasting consequences. *Bulletin of the American Mathematical Society*, 42(1):39–56, 2005.
- L. Xiong, X. Chen, T. Huang, J. Schneider, and J. Carbonell. Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In *SIAM International Conference on Data Mining*, 2010.
- C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation – An empirical study. *J. Mach. Learn. Res.*, 7:1887–1907, 2006.
- C. Yu and T. Joachims. Learning structural SVMs with latent variables. In *International Conference on Machine Learning (ICML)*, 2009.
- A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Comput.*, 15(4):915–936, 2003.