Hinge-Loss Markov Random Fields and Probabilistic Soft Logic

Stephen H. Bach* Matthias Broecheler† Bert Huang‡ Lise Getoor§

WORKING DRAFT—Version 2

Abstract

A fundamental challenge in developing high-impact machine learning 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 paper, we 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. We unite three approaches from the randomized algorithms, probabilistic graphical models, and fuzzy logic communities, showing that all three lead to the same inference objective. We 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 using a syntax based on first-order logic. We next introduce an algorithm for inferring most-probable variable assignments (MAP inference) that is much more scalable than general-purpose convex optimization software, because it uses message passing to take advantage of sparse dependency structures. We then show how to learn the parameters of HL-MRFs. The learned HL-MRFs are as accurate as analogous discrete models, but much more scalable. Together, these algorithms enable HL-MRFs and PSL to model rich, structured data at scales not previously possible.

1 Introduction

In many problems in 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. Machine learning subfields such as statistical relational learning (Getoor and Taskar, 2007), inductive logic programming (Muggleton and De Raedt, 1994), and structured prediction (BakIr et al., 2007) all seek to represent both the relational structure and dependencies in

*Computer Science Department, Stanford University. bach@cs.stanford.edu
†DataStax. matthias@datastax.com
‡Computer Science Department, Virginia Tech. bhuang@vt.edu
§Computer Science Department, University of California, Santa Cruz. getoor@soe.ucsc.edu

the data, and, with the ever-increasing size of available data, there is a growing need for models that are highly scalable while still able to capture rich structure.

In this paper, we introduce *hinge-loss Markov random fields* (HL-MRFs), a new class of probabilistic graphical models designed to enable scalable modeling of rich, structured data. HL-MRFs are analogous to discrete MRFs, undirected probabilistic graphical models in which probability mass is log-proportional to a weighted sum of feature functions. Unlike discrete MRFs, however, HL-MRFs are defined over continuous variables in the $[0, 1]$ unit interval, and their feature functions are hinge functions, so that probability density is lost according to a weighted sum of hinge losses. When designing classes of models, there is generally a trade off between scalability and expressivity: the more complex the types and connectivity structure of the dependencies, the more computationally challenging inference and learning become. HL-MRFs address a crucial gap between the two extremes. By using hinge-loss functions to model the dependencies among the variables, which admit highly scalable inference without restrictions on their connectivity structure, they can capture a very wide range of useful relationships. One reason they are so expressive is that hinge-loss dependencies are at the core of a number of scalable techniques for modeling both discrete and continuous structured data.

On the road to deriving HL-MRFs, we unify three different approaches to scalable inference in structured models: (1) randomized algorithms for MAX SAT (Goemans and Williamson, 1994), (2) local consistency relaxation (Wainwright and Jordan, 2008) for discrete Markov random fields defined using Boolean logic, and (3) reasoning about continuous information with fuzzy logic. We show that all three approaches lead to the same convex programming objective. We then derive HL-MRFs by generalizing this unified inference objective as a weighted sum of hinge-loss features and using them as the weighted features of graphical models. Since HL-MRFs generalize approaches to reasoning about relational data with weighted logical knowledge bases, they retain this high level of expressivity. As we show in Section 6.4, they are effective for modeling both discrete and continuous data.

We also introduce *probabilistic soft logic* (PSL), a new probabilistic programming language that makes HL-MRFs easy to define and use for large, relational data sets. This idea has been explored for other classes of models, such as Markov logic networks (Richardson and Domingos, 2006) for discrete MRFs, relational dependency networks (Neville and Jensen, 2007) for dependency networks, and probabilistic relational models (Getoor et al., 2002) for Bayesian networks. We build on these previous approaches, as well as the connection between hinge-loss potentials and logical clauses, to define PSL. In addition to probabilistic rules, PSL provides syntax that enables users to easily apply many common modeling techniques, such as domain and range constraints, blocking and canopy functions, and aggregate variables defined over other random variables.

Our next contribution is to introduce a number of inference and learning algorithms. First, we examine MAP inference, i.e., the problem of finding a most probable assignment to the unobserved random variables. MAP inference in HL-MRFs is always a convex optimization. Although any off-the-shelf optimization toolkit could be used, such methods typically do not leverage the sparse dependency structures common in graphical models. We introduce a consensus-optimization approach to MAP inference for HL-MRFs, showing how the problem can be decomposed using the alternating direction method of multipliers (ADMM) and how the resulting subproblems can be solved analytically for hinge-loss poten-

tials. Our approach enables HL-MRFs to easily scale beyond the capabilities of off-the-shelf optimization software or sampling-based inference in discrete MRFs. We then show how to learn HL-MRFs from training data using a variety of methods: maximum likelihood, maximum pseudolikelihood, and large margin estimation. Since maximum likelihood and large margin estimation rely on inference as subroutines, and maximum pseudolikelihood estimation is efficient by design, all of these methods are highly scalable for HL-MRFs. We evaluate them on core relational learning and structured prediction tasks, such as collective classification and link prediction. We show that HL-MRFs offer predictive accuracy comparable to state-of-the-art discrete models while scaling dramatically better to large data sets.

This paper brings together and expands a range of work on scalable models for structured data that can be either discrete, continuous, or both. In a previous publication, we proposed probabilistic similarity logic (Broecheler et al., 2010a), a probabilistic programming language for reasoning about similarity in structured data. We also proposed HL-MRFs, which generalize the kind of models that can be defined with probabilistic similarity logic, as well as ADMM inference for them (Bach et al., 2012, 2013). Next, we showed that the hinge-loss potentials of HL-MRFs can also be motivated by two different approximate inference methods for discrete models: randomized algorithms for MAX SAT and local consistency relaxation (Bach et al., 2015). Now, in this paper, we present the full motivation for and derivation of HL-MRFs, including unifying three approaches to scalable inference. We also present the PSL language, algorithms for inference and learning, and empirical results demonstrating that HL-MRFs and PSL are effective for modeling both discrete and continuous data.

The effectiveness of HL-MRFs and PSL has also been demonstrated in other publications 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 indicates HL-MRFs and PSL directly address an open need of the machine learning community.

The paper is organized as follows. In Section 2, we first consider models for structured prediction that are defined using logical clauses. We unify three different approaches to scalable inference in such models, showing that they all optimize the same convex objective. We then generalize this objective in Section 3 to derive HL-MRFs. In Section 4, we introduce PSL, giving both a precise specification of the language and many examples of common usage. Next we introduce a very scalable message-passing algorithm for MAP inference in Section 5 and a number of learning algorithms in Section 6, evaluating them on a range of tasks for structured data. Finally, in Section 7, we discuss related work.

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

$$P(\boldsymbol{x}) \propto \exp\left(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x})\right) \ . \tag{1}$$

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 $\boldsymbol{C} = \{C_1, \ldots, C_m\}$ where each clause $C_j \in \boldsymbol{C}$ is a disjunction of literals and each literal is a variable x or its negation $\neg x$ drawn from the variables $\boldsymbol{x}$ such that each variable $x_i \in \boldsymbol{x}$ appears at most once in C_j. Let I_j^+ (resp. I_j^-) $\subset \{1, \ldots, n\}$ be the set of indices of the variables that are not negated (resp. negated) in C_j. Then C_j can be written as

$$\left(\bigvee_{i \in I_j^+} x_i\right) \bigvee \left(\bigvee_{i \in I_j^-} \neg x_i\right) \ . \tag{2}$$

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

$$\bigwedge_{i \in I_j^-} x_i \implies \bigvee_{i \in I_j^+} x_i \ . \tag{3}$$

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 $\boldsymbol{C}$ 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 $\boldsymbol{x}$ satisfies C_j, then we let $\phi_j(\boldsymbol{x})$ equal 1, and we let it equal 0 otherwise. For our subsequent analysis we now let $w_j \geq 0$ $(\forall j = 1, \ldots, m)$. The resulting MRF preserves the structured dependencies described in $\boldsymbol{C}$, 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 $\boldsymbol{C}$. 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 defined by $\boldsymbol{C}$ is the integer linear program

$$\begin{aligned} \arg\max_{\boldsymbol{x} \in \{0,1\}^n} P(\boldsymbol{x}) &\equiv \arg\max_{\boldsymbol{x} \in \{0,1\}^n} \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) \\ &\equiv \arg\max_{\boldsymbol{x} \in \{0,1\}^n} \sum_{C_j \in \boldsymbol{C}} w_j \min \left\{ \sum_{i \in I_j^+} x_i + \sum_{i \in I_j^-} (1 - x_i), 1 \right\} . \end{aligned} \tag{4}$$

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 2.1 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 2.2 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 2.3, we show that the unified inference objective is also equivalent

[1] Markov 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.

to exact MAP inference if the knowledge base $\boldsymbol{C}$ 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 3.1 to derive hinge-loss MRFs, and in the rest of the paper 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.

2.1 MAX SAT Relaxation

One approach to approximating objective (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 (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

$$\hat{w}_j = w_j \left(1 - \prod_{i \in I_j^+} (1 - p_i) \prod_{i \in I_j^-} p_i \right) , \tag{5}$$

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

$$\hat{W} = \sum_{C_j \in \boldsymbol{C}} w_j \left(1 - \prod_{i \in I_j^+} (1 - p_i) \prod_{i \in I_j^-} p_i \right) . \tag{6}$$

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 $\hat{\boldsymbol{y}}^\star$ be the optimum to the linear program

$$\underset{\hat{\boldsymbol{y}} \in [0,1]^n}{\arg\max} \sum_{C_j \in \boldsymbol{C}} w_j \min \left\{ \sum_{i \in I_j^+} \hat{y}_i + \sum_{i \in I_j^-} (1 - \hat{y}_i), 1 \right\} . \tag{7}$$

Observe that objectives (4) and (7) are of the same form, except that the variables are relaxed to the unit hypercube in objective (7). Goemans and Williamson (1994) showed

that if p_i is set to $\hat{y}_i^\star$ for all i, then $\hat{W} \geq .632\ Z^\star$, where $Z^\star$ 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

$$p_i = \frac{1}{2}\hat{y}_i^\star + \frac{1}{4}\ . \tag{8}$$

In this way, objective (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 (7) and function (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 $\hat{W}$ (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.

2.2 Local Consistency Relaxation

Another approach to approximating objective (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 $\phi_j \in \boldsymbol{\phi}$, let $\boldsymbol{\theta}_j$ be a marginal distribution over joint assignments $\boldsymbol{x}_j$. For example, $\theta_j(\boldsymbol{x}_j)$ is the probability that the subset of variables associated with potential ϕ_j is in a particular joint state $\boldsymbol{x}_j$. Also, let $x_j(i)$ denote the setting of the variable with index i in the state $\boldsymbol{x}_j$.

With this variational formulation, inference can be relaxed to an optimization over the *first-order local polytope* $\mathbb{L}$. Let $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$ 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

$$\mathbb{L} \triangleq \left\{ (\boldsymbol{\theta}, \boldsymbol{\mu}) \geq \mathbf{0} \;\middle|\; \begin{array}{ll} \sum_{\boldsymbol{x}_j | x_j(i)=k} \theta_j(\boldsymbol{x}_j) = \mu_i(k) & \forall i, j, k \\ \sum_{\boldsymbol{x}_j} \theta_j(\boldsymbol{x}_j) = 1 & \forall j \\ \sum_{k=0}^{K_i - 1} \mu_i(k) = 1 & \forall i \end{array} \right\}, \tag{9}$$

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

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

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

$$\arg\max_{(\boldsymbol{\theta},\boldsymbol{\mu})\in\mathbb{L}} \sum_{j=1}^{m} w_j \sum_{\boldsymbol{x}_j} \theta_j(\boldsymbol{x}_j)\ \phi_j(\boldsymbol{x}_j), \tag{10}$$

which is an upper bound on the true MAP objective. Much work has focused on solving the first-order local consistency relaxation for large-scale MRFs, which we discuss further in Section 7. 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 $\boldsymbol{C}$ 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 $\boldsymbol{\mu}^\star$ of objective (10) is an optimum $\hat{\boldsymbol{y}}^\star$ of objective (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.

2.3 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 $\boldsymbol{C}$ 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 $\boldsymbol{x}$ 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 $\boldsymbol{x}$ 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 $\neg$):

$$x_1 \wedge x_2 = \max\{x_1 + x_2 - 1, 0\} \tag{11}$$
$$x_1 \vee x_2 = \min\{x_1 + x_2, 1\} \tag{12}$$
$$\neg x = 1 - x \ . \tag{13}$$

The analogous MAX SAT problem for Łukasiewicz logic is therefore

$$\operatorname*{arg\,max}_{\boldsymbol{x} \in [0,1]^n} \sum_{C_j \in \boldsymbol{C}} w_j \min\left\{ \sum_{i \in I_j^+} x_i + \sum_{i \in I_j^-} (1 - x_i), 1 \right\} \ , \tag{14}$$

which is identical in form to objective (7). Therefore, if an MRF is defined over continuous variables with domain $[0, 1]^n$ and the logical knowledge base $\boldsymbol{C}$ 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 Hinge-Loss Markov Random Fields

We have shown that a single convex program can be used to reason scalably and accurately about both discrete and continuous information. In this section, we generalize this inference objective to derive *hinge-loss Markov random fields* (HL-MRFs), a new kind of probabilistic graphical model. HL-MRFs preserve the convexity (during MAP inference) and expressivity of the logic-based objective, but additionally support an even richer space of dependencies. To begin, we define HL-MRFs as density functions over continuous variables $\boldsymbol{y} = (y_1, \ldots, y_n)$ 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 2, 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.

3.1 Derivation

To derive HL-MRFs, we will generalize the unified inference objective of Section 2 in several ways, which we restate for our semantics-agnostic variables:

$$\operatorname*{arg\,max}_{\boldsymbol{y} \in [0,1]^n} \sum_{C_j \in \boldsymbol{C}} w_j \min\left\{ \sum_{i \in I_j^+} y_i + \sum_{i \in I_j^-} (1 - y_i), 1 \right\} \ . \tag{15}$$

For now, we are still assuming that the objective terms are defined using a weighted knowledge base $\boldsymbol{C}$, 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:

$$\arg\min_{\boldsymbol{y}\in[0,1]^n} \sum_{C_j\in\boldsymbol{C}} w_j \max\left\{1 - \sum_{i\in I_j^+} y_i - \sum_{i\in I_j^-}(1-y_i), 0\right\} , \tag{16}$$

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

$$1 - \sum_{i\in I_j^+} y_i - \sum_{i\in I_j^-}(1-y_i) \leq 0 \tag{17}$$

is from being satisfied.

3.1.1 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 $\ell_j(\boldsymbol{y})$ that is linear in $\boldsymbol{y}$. Then, the new inference objective is

$$\arg\min_{\boldsymbol{y}\in[0,1]^n} \sum_{j=1}^{m} w_j \max\left\{\ell_j(\boldsymbol{y}), 0\right\} . \tag{18}$$

Now each term represents the distance to satisfaction of a linear constraint $\ell_j(\boldsymbol{y}) \leq 0$. 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, $\ell_j(\boldsymbol{y}) \leq 0$ and $-\ell_j(\boldsymbol{y}) \leq 0$, can be combined to represent a relaxed equality constraint $\ell_j(\boldsymbol{y}) = 0$.

3.1.2 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.

3.1.3 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 $\ell_j(\boldsymbol{y}) = 0$. 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:

$$\arg\min_{y_1 \in [0,1]} \; w_1 \max\{y_1, 0\} + w_2 \max\{1 - y_1, 0\} \;\; . \tag{19}$$

If $w_1 > w_2 \geq 0$, then the optimizer is $y_1 = 0$ because the term that prefers $y_1 = 0$ overrules the term 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 the conflicting preferences. 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

$$\arg\min_{y_1 \in [0,1]} \; w_1 \left(\max\{y_1, 0\}\right)^2 + w_2 \left(\max\{1 - y_1, 0\}\right)^2 \tag{20}$$

the optimizer is now $y_1 = \frac{w_2}{w_1 + w_2}$, 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

$$\begin{aligned} \arg\min_{(y_1, y_2) \in [0,1]^2} \quad & \max\{0.9 - y_1, 0\} + \max\{0.6 - y_2, 0\} \\ \text{such that} \quad & y_1 + y_2 \leq 1 \;\; . \end{aligned} \tag{21}$$

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

$$\begin{aligned} \underset{(y_1,y_2)\in[0,1]^2}{\arg\min} \quad & \left(\max\left\{0.9 - y_1, 0\right\}\right)^2 + \left(\max\left\{0.6 - y_2, 0\right\}\right)^2 \\ \text{such that} \quad & y_1 + y_2 \leq 1 \ , \end{aligned} \tag{22}$$

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.

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

$$\phi_j(\boldsymbol{y}, \boldsymbol{x}) = \left(\max\left\{\ell_j(\boldsymbol{y}, \boldsymbol{x}), 0\right\}\right)^{p_j} \tag{23}$$

where ℓ_j is a linear function of $\boldsymbol{y}$ and $\boldsymbol{x}$ and $p_j \in \{1, 2\}$. Let $\boldsymbol{c} = (c_1, \ldots, c_r)$ 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

$$\tilde{\boldsymbol{D}} = \left\{ (\boldsymbol{y}, \boldsymbol{x}) \in \boldsymbol{D} \;\middle|\; \begin{array}{l} c_k(\boldsymbol{y}, \boldsymbol{x}) = 0, \forall k \in \mathcal{E} \\ c_k(\boldsymbol{y}, \boldsymbol{x}) \leq 0, \forall k \in \mathcal{I} \end{array} \right\} \ . \tag{24}$$

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

$$f_{\boldsymbol{w}}(\boldsymbol{y}, \boldsymbol{x}) = \sum_{j=1}^{m} w_j \phi_j(\boldsymbol{y}, \boldsymbol{x}) \ . \tag{25}$$

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* $\boldsymbol{y}$ *and conditioned on random variables* $\boldsymbol{x}$ *is a probability density defined as follows: if* $(\boldsymbol{y}, \boldsymbol{x}) \notin \tilde{\boldsymbol{D}}$*, then* $P(\boldsymbol{y}|\boldsymbol{x}) = 0$*; if* $(\boldsymbol{y}, \boldsymbol{x}) \in \tilde{\boldsymbol{D}}$*, then*

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{w}, \boldsymbol{x})} \exp\left(-f_{\boldsymbol{w}}(\boldsymbol{y}, \boldsymbol{x})\right) \tag{26}$$

where

$$Z(\boldsymbol{w}, \boldsymbol{x}) = \int_{\boldsymbol{y}|(\boldsymbol{y},\boldsymbol{x})\in\tilde{\boldsymbol{D}}} \exp\left(-f_{\boldsymbol{w}}(\boldsymbol{y}, \boldsymbol{x})\right) \, d\boldsymbol{y} \ . \tag{27}$$

In the rest of this paper, 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.

4 Probabilistic Soft Logic

In this section 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 2. Arithmetic rules provide additional syntax for defining an even wider range of hinge-loss potentials and hard constraints.

4.1 Definition

In this subsection 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 $\{\texttt{A}, \ldots, \texttt{Z}, \texttt{a}, \ldots, \texttt{z}\}$), 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.[3] 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 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.*

[3] Note 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.

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 $\mathcal{O} : \mathcal{A} \to [0, 1] \cup \{\emptyset\}$ 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 paper, 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 section 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:

```
Advises("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,

$$\mathcal{A} = \left\{ \begin{array}{l} \texttt{Category("d1", "politics")}, \\ \texttt{Category("d1", "sports")}, \\ \texttt{Category("d2", "politics")}, \\ \texttt{Category("d2", "sports")} \end{array} \right\} . \tag{28}$$

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 `!=`. 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 != 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 $P(\boldsymbol{y}|\boldsymbol{x})$ 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 $\boldsymbol{y}$, and otherwise it is included in the observations $\boldsymbol{x}$ 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 `~` 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 disjuncted with the head. For example

```
   P1(A, B) && P2(A, B) -> P3(A, B) || P4(A,B)
≡ !(P1(A, B) && P2(A, B)) || P3(A, B) || P4(A,B)
≡ !P1(A, B) || !P2(A, B) || P3(A, B) || P4(A,B)
```

Therefore, any formula written as an implication with a literal or conjunction or 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 (`^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

```
Friends(X, Y) && Friends(Y, Z) -> 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 $\boldsymbol{x}$. 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 2. 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

$$1 - \sum_{i \in I^+} y_i - \sum_{i \in I^-} (1 - y_i) \leq 0 \ . \tag{29}$$

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

$$\phi(\boldsymbol{y}, \boldsymbol{x}) = \max \left\{ 1 - \sum_{i \in I^+} y_i - \sum_{i \in I^-} (1 - y_i), 0 \right\} \tag{30}$$

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

$$\phi(\boldsymbol{y}, \boldsymbol{x}) = \left(\max \left\{ 1 - \sum_{i \in I^+} y_i - \sum_{i \in I^-} (1 - y_i), 0 \right\} \right)^2 \tag{31}$$

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

$$c(\boldsymbol{y}, \boldsymbol{x}) = 1 - \sum_{i \in I^+} y_i - \sum_{i \in I^-} (1 - y_i) \tag{32}$$

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

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

```
3 :  Friends(A, B) && Friends(B, C) -> Friends(C, A) ^2
```

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

$$\mathcal{A} = \left\{ \begin{array}{l} \texttt{Friends("p1", "p2")}, \\ \texttt{Friends("p1", "p3")}, \\ \texttt{Friends("p2", "p1")}, \\ \texttt{Friends("p2", "p3")}, \\ \texttt{Friends("p3", "p1")}, \\ \texttt{Friends("p3", "p2")} \end{array} \right\} , \tag{33}$$

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

```
3 :  Friends("p1", "p2") && Friends("p2", "p3") -> Friends("p3", "p1") ^2
```

which is equivalent to

```
3 :  !Friends("p1", "p2") || !Friends("p2", "p3") || Friends("p3", "p1") ^2
```

If the atoms `Friends("p1", "p2")`, `Friends("p2", "p3")`, and `Friends("p3", "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

$$3 \ (\max\{y_1 + y_2 - y_3 - 1, 0\})^2 \ . \tag{34}$$

Since the grounding process uses the mapping from Section 2, 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:

```
Liberal(P) + 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 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 $c(\boldsymbol{y}, \boldsymbol{x})$ 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

```
Friends(P, +F)
```

is a placeholder for the sum of all ground atoms in $\mathcal{A}$ that have a given first argument. Note that sum variables can be used at most once in a rule, i.e., each sum variable in a rule must have a unique identifier. 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

```
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`.

```
Link(X, +Y) <= 1 .
```

Then, we can add the following select statement:

```
{Y: Property(Y)}
```

Then, the hard linear constraints templated by the arithmetic rule will only sum over constants substituted for `Y` such that `Property(Y)` is non-zero.

In arithmetic rules, atoms can also be modified with coefficients. These coefficients can be hard-coded. As a simple example, in the rule

```
Susceptible(X) >= 0.5 Biomarker1(X) + 0.5 Biomarker2(X) .
```

the property `Susceptible/1`, which represents the degree to which a patient is susceptible to a particular disease, must be at least the average value of two biomarkers.

PSL also supports two pieces of 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 functions enable rules that depend on the number of substitutions in order to be scaled correctly, such as averaging. Cardinality is denoted by enclosing a sum variable, without the +, in pipes. For example, the rule

```
1 / |Y| Friends(X, +Y) = Friendliness(X) .
```

defines the `Friendliness/1` property of a person `X` in a social network as the average strength of their outgoing friendship links. In cases in which `Friends/2` is not symmetric, we can extend this rule to sum over both outgoing and incoming links:

```
1 / |Y1| |Y2| Friends(X, +Y1) + 1 / |Y1| |Y2| Friends(+Y2, X)
= Friendliness(X) .
```

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 for matching two sets of constants requires that the sum of the `Matched/2` atoms be the minimum of the sizes of the two sets:

```
Matched(+X, +Y) = @Min[|X|, |Y|] .
```

Note that PSL's coefficient syntax can also be used to define constants, as in this example.

So far we have focused on using arithmetic rules to define templates for linear constraints, but they can also be used to define hinge-loss potentials. For example, the following arithmetic rule prefers that the degree to which a person `X` is extroverted (represented with `Extroverted/1`) does not exceed the average extroversion of their friends:

```
10 :  Extroverted(X) <= 1 / |Y| Extroverted(+Y) ^2
{Y: Friends(X, Y) || Friends(Y, X)}
```

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

$$10 \left(\max \left\{ y_{i'} - \frac{1}{|\mathcal{F}|} \sum_{i \in \mathcal{F}} y_i, 0 \right\} \right)^2 \tag{35}$$

where $y_{i'}$ is the variable corresponding to a grounding of the atom `Extroverted(X)` and $\mathcal{F}$ is the set of the indices of the variables corresponding to `Extroverted(Y)` atoms of the friends `Y` that satisfy the rule's select statement. Note that the weight of 10 is distinct from the coefficients in the linear constraint $\ell(\boldsymbol{y}, \boldsymbol{x}) \leq 0$ 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 $\ell(\boldsymbol{y}, \boldsymbol{x}) \leq 0$ and one using $-\ell(\boldsymbol{y}, \boldsymbol{x}) \leq 0$. 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. Each sum variable in an arithmetic rule must have a unique identifier. 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 $c(\boldsymbol{y}, \boldsymbol{x}) \leq 0$. Then $c(\boldsymbol{y}, \boldsymbol{x})$ 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 $c(\boldsymbol{y}, \boldsymbol{x}) = 0$, $c(\boldsymbol{y}, \boldsymbol{x})$ 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 $\ell(\boldsymbol{y}, \boldsymbol{x}) \leq 0$ and included as a potential of the form

$$\phi(\boldsymbol{y}, \boldsymbol{x}) = \max\{\ell(\boldsymbol{y}, \boldsymbol{x}), 0\} \tag{36}$$

with a weight of w. If the arithmetic rule is a weighted equality with weight w, each ground rule is again manipulated to $\ell(\boldsymbol{y}, \boldsymbol{x}) \leq 0$ and two potentials are included,

$$\phi_1(\boldsymbol{y}, \boldsymbol{x}) = \max\{\ell(\boldsymbol{y}, \boldsymbol{x}), 0\}, \quad \phi_2(\boldsymbol{y}, \boldsymbol{x}) = \max\{-\ell(\boldsymbol{y}, \boldsymbol{x}), 0\}\ , \tag{37}$$

each with a weight of w. In either case, if the weighted arithmetic rule is annotated with `^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

$$-w \quad : \quad \left(\bigvee_{i \in I_j^+} x_i \right) \bigvee \left(\bigvee_{i \in I_j^-} \neg x_i \right) \quad . \tag{38}$$

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:

$$w \quad : \quad \left(\bigwedge_{i \in I_j^+} \neg x_i \right) \bigwedge \left(\bigwedge_{i \in I_j^-} x_i \right) \quad . \tag{39}$$

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 (38) is used to define a potential in a discrete MRF, replacing it with a potential defined by (39) 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

$$w_1 \quad : \quad x_1 \wedge x_2 \tag{40}$$

with three disjunctions

$$w_2 \quad : \quad x_1 \vee x_2 \tag{41}$$

$$w_2 \quad : \quad \neg x_1 \vee x_2 \tag{42}$$

$$w_2 \quad : \quad x_1 \vee \neg x_2 \tag{43}$$

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 (38) and (39) 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

```
Label(Document, +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

```
Advisor(+Professor, 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:

```
Same(Person1, +Person2) <= 1 .
Same(+Person1, Person2) <= 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 :  !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) = 1 / |F| Interest(+F, I) .
{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):

```
SameFriends(A, B) = 1 / @Max[|FA|, |FB|] SamePerson(+FA, +FB) .
{FA : Friends(A, FA)}
{FB : Friends(B, FB)}
SamePerson(+P1, P2) = 1 .
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.

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 $\boldsymbol{y}$ given

observations $\boldsymbol{x}$. In HL-MRFs, the normalizing function $Z(\boldsymbol{w}, \boldsymbol{x})$ is constant over $\boldsymbol{y}$ and the exponential is maximized by minimizing its negated argument, so the MAP problem is

$$\begin{aligned} \arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}) &\equiv \arg\min_{\boldsymbol{y}} f_{\boldsymbol{w}}(\boldsymbol{y}, \boldsymbol{x}) \\ &\equiv \arg\min_{\boldsymbol{y} \in [0,1]^n} \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{y}, \boldsymbol{x}) \\ \text{such that} \quad & c_k(\boldsymbol{y}, \boldsymbol{x}) = 0, \quad \forall k \in \mathcal{E} \\ & c_k(\boldsymbol{y}, \boldsymbol{x}) \leq 0, \quad \forall k \in \mathcal{I} \ . \end{aligned} \tag{44}$$

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 $f_{\boldsymbol{w}}$ 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 $P(\boldsymbol{y}|\boldsymbol{x})$, 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 $\boldsymbol{y}_{(L,j)}$ be a copy of the variables in $\boldsymbol{y}$ that are used in the potential function ϕ_j, $j = 1, \ldots, m$ and $\boldsymbol{y}_{(L,k+m)}$ be a copy of those used in the constraint function c_k, $k = 1, \ldots, r$. We refer to the concatenation of all of these vectors as $\boldsymbol{y}_L$. We also introduce an indicator function I_k for each constraint function where $I_k \left[c_k(\boldsymbol{y}_{(L,k+m)}, \boldsymbol{x}) \right] = 0$ 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 $\boldsymbol{y}$, 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 $\boldsymbol{y}_{(C,\hat{i})}$ be the variables in $\boldsymbol{y}$ that correspond to $\boldsymbol{y}_{(L,\hat{i})}$, $\hat{i} = 1, \ldots, m + r$. Operators between $\boldsymbol{y}_{(L,\hat{i})}$ and $\boldsymbol{y}_{(C,\hat{i})}$ are defined element-wise, pairing the corresponding

copied variables. Consensus optimization solves the reformulated MAP problem

$$\begin{aligned} \underset{(\boldsymbol{y}_L, \boldsymbol{y})}{\arg\min} \quad & \sum_{j=1}^{m} w_j \phi_j \left(\boldsymbol{y}_{(L,j)}, \boldsymbol{x} \right) + \sum_{k=1}^{r} I_k \left[c_k \left(\boldsymbol{y}_{(L,k+m)}, \boldsymbol{x} \right) \right] + \sum_{i=1}^{n} I_{[0,1]} \left[y_i \right] \\ \text{such that} \quad & \boldsymbol{y}_{(L,\hat{i})} = \boldsymbol{y}_{(C,\hat{i})} \quad \forall \hat{i} = 1, \ldots, m+r \ . \end{aligned} \tag{45}$$

Inspection shows that problems (44) and (45) are equivalent.

This reformulation enables us to relax the equality constraints $\boldsymbol{y}_{(L,\hat{i})} = \boldsymbol{y}_{(C,\hat{i})}$ in order to divide problem (45) 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 $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{m+r})$ be a concatenation of vectors of Lagrange multipliers. Then the augmented Lagrangian is

$$\begin{aligned} \mathcal{L}(\boldsymbol{y}_L, \boldsymbol{\alpha}, \boldsymbol{y}) = & \sum_{j=1}^{m} w_j \phi_j \left(\boldsymbol{y}_{(L,j)}, \boldsymbol{x} \right) + \sum_{k=1}^{r} I_k \left[c_k \left(\boldsymbol{y}_{(L,k+m)}, \boldsymbol{x} \right) \right] + \sum_{i=1}^{n} I_{[0,1]} \left[y_i \right] \\ & + \sum_{\hat{i}=1}^{m+r} \boldsymbol{\alpha}_{\hat{i}}^{\top} \left(\boldsymbol{y}_{(L,\hat{i})} - \boldsymbol{y}_{(C,\hat{i})} \right) + \frac{\rho}{2} \sum_{\hat{i}=1}^{m+r} \left\| \boldsymbol{y}_{(L,\hat{i})} - \boldsymbol{y}_{(C,\hat{i})} \right\|_2^2 \end{aligned} \tag{46}$$

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

$$\boldsymbol{\alpha}_{\hat{i}}^{t} \leftarrow \boldsymbol{\alpha}_{\hat{i}}^{t-1} + \rho \left(\boldsymbol{y}_{(L,\hat{i})}^{t-1} - \boldsymbol{y}_{(C,\hat{i})}^{t-1} \right) \qquad \forall \hat{i} = 1, \ldots, m+r \tag{47}$$

$$\boldsymbol{y}_L^{t} \leftarrow \underset{\boldsymbol{y}_L}{\arg\min} \ \mathcal{L} \left(\boldsymbol{y}_L, \boldsymbol{\alpha}^t, \boldsymbol{y}^{t-1} \right) \tag{48}$$

$$\boldsymbol{y}^{t} \leftarrow \underset{\boldsymbol{y}}{\arg\min} \ \mathcal{L} \left(\boldsymbol{y}_L^t, \boldsymbol{\alpha}^t, \boldsymbol{y} \right) \tag{49}$$

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

$$\|\bar{\boldsymbol{r}}^t\|_2 \triangleq \left(\sum_{\hat{i}=1}^{m+r} \|\boldsymbol{y}_{(L,\hat{i})}^t - \boldsymbol{y}_{(C,\hat{i})}^t\|_2^2 \right)^{\frac{1}{2}} \qquad \|\bar{\boldsymbol{s}}^t\|_2 \triangleq \rho \left(\sum_{i=1}^{n} \mathcal{K}_i (y_i^t - y_i^{t-1})^2 \right)^{\frac{1}{2}} \tag{50}$$

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

$$\|\bar{\boldsymbol{r}}^t\|_2 \leq \epsilon^{\text{abs}} \sqrt{\sum_{i=1}^{n} \mathcal{K}_i} + \epsilon^{\text{rel}} \max \left\{ \left(\sum_{\hat{i}=1}^{m+r} \|\boldsymbol{y}_{(L,\hat{i})}^t\|_2^2 \right)^{\frac{1}{2}}, \left(\sum_{i=1}^{n} \mathcal{K}_i (y_i^t)^2 \right)^{\frac{1}{2}} \right\} \tag{51}$$

$$\|\bar{\boldsymbol{s}}^t\|_2 \leq \epsilon^{\text{abs}} \sqrt{\sum_{i=1}^{n} \mathcal{K}_i} + \epsilon^{\text{rel}} \left(\sum_{\hat{i}=1}^{m+r} \|\boldsymbol{\alpha}_{\hat{i}}^t\|_2^2 \right)^{\frac{1}{2}} \tag{52}$$

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

5.2 Block Updates

We now describe how to implement the ADMM block updates (47), (48), and (49). Updating the Lagrange multipliers $\boldsymbol{\alpha}$ is a simple step in the gradient direction (47). Updating the local copies $\boldsymbol{y}_L$ (48) decomposes over each potential and constraint in the HL-MRF. For the variables $\boldsymbol{y}_{(L,j)}$ for each potential ϕ_j, this requires independently optimizing the weighted potential plus a squared norm:

$$\underset{\boldsymbol{y}_{(L,j)}}{\arg\min} \; w_j \left(\max \left\{ \ell_j(\boldsymbol{y}_{(L,j)}, \boldsymbol{x}), 0 \right\} \right)^{p_j} + \frac{\rho}{2} \left\| \boldsymbol{y}_{(L,j)} - \boldsymbol{y}_{(C,j)} + \frac{1}{\rho} \boldsymbol{\alpha}_j \right\|_2^2 . \tag{53}$$

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 $\boldsymbol{y}_{(L,j)}^{\star}$, the optimizer of problem (53), which correspond to the three regions in which the solution must lie: (1) the region $\ell(\boldsymbol{y}_{(L,j)}, \boldsymbol{x}) < 0$, (2) the region $\ell(\boldsymbol{y}_{(L,j)}, \boldsymbol{x}) > 0$, and (3) the region $\ell(\boldsymbol{y}_{(L,j)}, \boldsymbol{x}) = 0$. 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 $\boldsymbol{y}_{(C,j)} - \boldsymbol{\alpha}_j/\rho$. If $\ell_j(\boldsymbol{y}_{(C,j)} - \boldsymbol{\alpha}_j/\rho, \boldsymbol{x}) \leq 0$, then $\boldsymbol{y}_{(L,j)}^{\star} = \boldsymbol{y}_{(C,j)} - \boldsymbol{\alpha}_j/\rho$, because it optimizes both the potential and the squared norm independently. If instead $\ell_j(\boldsymbol{y}_{(C,j)} - \boldsymbol{\alpha}_j/\rho, \boldsymbol{x}) > 0$, then we can conclude that $\ell_j(\boldsymbol{y}_{(L,j)}^{\star}, \boldsymbol{x}) \geq 0$, 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 $\boldsymbol{y}_{(L,j)}$, setting the gradient equal to the zero vector, and solving for $\boldsymbol{y}_{(L,j)}$. In other words, the optimizer is the solution for $\boldsymbol{y}_{(L,j)}$ to the equation

$$g \nabla_{\boldsymbol{y}_{(L,j)}} \left[w_j \left(\ell_j(\boldsymbol{y}_{(L,j)}, \boldsymbol{x}) \right)^{p_j} + \frac{\rho}{2} \left\| \boldsymbol{y}_{(L,j)} - \boldsymbol{y}_{(C,j)} + \frac{1}{\rho} \boldsymbol{\alpha}_j \right\|_2^2 \right] = \mathbf{0} . \tag{54}$$

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 $\boldsymbol{y}'_{(L,j)}$ be the optimizer of the modified problem, i.e., the solution to equation (54). If $\ell_j(\boldsymbol{y}'_{(L,j)}, \boldsymbol{x}) \geq 0$, then $\boldsymbol{y}^\star_{(L,j)} = \boldsymbol{y}'_{(L,j)}$ because we know the solution lies in the region $\ell_j(\boldsymbol{y}_{(L,j)}, \boldsymbol{x}) \geq 0$ and the objective of problem (53) and the modified objective are equal on that region. In fact, if $p_j = 2$, then $\ell_j(\boldsymbol{y}'_{(L,j)}, \boldsymbol{x}) \geq 0$ whenever $\ell_j(\boldsymbol{y}_{(C,j)} - \boldsymbol{\alpha}_j/\rho, \boldsymbol{x}) \geq 0$, because the modified term is symmetric about the line $\ell_j(\boldsymbol{y}_{(L,j)}, \boldsymbol{x}) = 0$. We therefore will only reach the following third case when $p_j = 1$. If $\ell_j(\boldsymbol{y}_{(C,j)} - \boldsymbol{\alpha}_j/\rho, \boldsymbol{x}) > 0$ and $\ell_j(\boldsymbol{y}'_{(L,j)}, \boldsymbol{x}) < 0$, then we can conclude that $\boldsymbol{y}^\star_{(L,j)}$ is the projection of $\boldsymbol{y}_{(C,j)} - \boldsymbol{\alpha}_j/\rho$ onto the hyperplane $c_k(\boldsymbol{y}_{(L,j)}, \boldsymbol{x}) = 0$. 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 (53) reduces to the projection operation.

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

$$\underset{\boldsymbol{y}_{(L,k+m)}}{\arg\min} \; I_k \left[c_k(\boldsymbol{y}_{(L,k+m)}, \boldsymbol{x}) \right] + \frac{\rho}{2} \left\| \boldsymbol{y}_{(L,k+m)} - \boldsymbol{y}_{(C,k+m)} + \frac{1}{\rho}\boldsymbol{\alpha}_{k+m} \right\|_2^2 . \tag{55}$$

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

To update the variables $\boldsymbol{y}$ (49), we solve the optimization

$$\underset{\boldsymbol{y}}{\arg\min} \; \sum_{i=1}^{n} I_{[0,1]}\left[y_i\right] + \frac{\rho}{2} \sum_{\hat{i}=1}^{m+r} \left\| \boldsymbol{y}_{(L,\hat{i})} - \boldsymbol{y}_{(C,\hat{i})} + \frac{1}{\rho}\boldsymbol{\alpha}_{\hat{i}} \right\|_2^2 . \tag{56}$$

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 $\texttt{copies}(y_i)$ 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

$$y_i \leftarrow \frac{1}{|\texttt{copies}(y_i)|} \sum_{y_c \in \texttt{copies}(y_i)} \left(y_c + \frac{\alpha_c}{\rho} \right) \tag{57}$$

and clip the result to $[0, 1]$. Specifically, if, after update (57), $y_i > 1$, 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 (47), (48), and (49). In the pseudocode, we have interleaved updates (47)

Algorithm 1 MAP Inference for HL-MRFs

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

Initialize $\boldsymbol{y}_{(L,j)}$ as local copies of variables $\boldsymbol{y}_{(C,j)}$ that are in ϕ_j, $j = 1, \ldots, m$
Initialize $\boldsymbol{y}_{(L,k+m)}$ as local copies of variables $\boldsymbol{y}_{(C,k+m)}$ that are in c_k, $k = 1, \ldots, r$
Initialize Lagrange multipliers $\boldsymbol{\alpha}_{\hat{i}}$ corresponding to copies $\boldsymbol{y}_{(L,\hat{i})}$, $\hat{i} = 1, \ldots, m + r$

while not converged **do**

  for $j = 1, \ldots, m$ **do**
    $\boldsymbol{\alpha}_j \leftarrow \boldsymbol{\alpha}_j + \rho(\boldsymbol{y}_{(L,j)} - \boldsymbol{y}_{(C,j)})$
    $\boldsymbol{y}_{(L,j)} \leftarrow \boldsymbol{y}_{(C,j)} - \frac{1}{\rho}\boldsymbol{\alpha}_j$
    if $\ell_j(\boldsymbol{y}_{(L,j)}, \boldsymbol{x}) > 0$ **then**
      $\boldsymbol{y}_{(L,j)} \leftarrow \arg\min_{\boldsymbol{y}_{(L,j)}} w_j \left(\ell_j(\boldsymbol{y}_{(L,j)}, \boldsymbol{x})\right)^{p_j} + \frac{\rho}{2} \left\| \boldsymbol{y}_{(L,j)} - \boldsymbol{y}_{(C,j)} + \frac{1}{\rho}\boldsymbol{\alpha}_j \right\|_2^2$
      if $\ell_j(\boldsymbol{y}_{(L,j)}, \boldsymbol{x}) < 0$ **then**
        $\boldsymbol{y}_{(L,j)} \leftarrow \text{Proj}_{\ell_j=0}(\boldsymbol{y}_{(C,j)} - \frac{1}{\rho}\boldsymbol{\alpha}_j)$
      end if
    end if
  end for

  for $k = 1, \ldots, r$ **do**
    $\boldsymbol{\alpha}_{k+m} \leftarrow \boldsymbol{\alpha}_{k+m} + \rho(\boldsymbol{y}_{(L,k+m)} - \boldsymbol{y}_{(C,k+m)})$
    $\boldsymbol{y}_{(L,k+m)} \leftarrow \text{Proj}_{c_k}(\boldsymbol{y}_{(C,k+m)} - \frac{1}{\rho}\boldsymbol{\alpha}_{k+m})$
  end for

  for $i = 1, \ldots, n$ **do**
    $y_i \leftarrow \frac{1}{|\texttt{copies}(y_i)|} \sum_{y_c \in \texttt{copies}(y_i)} \left(y_c + \frac{\alpha_c}{\rho}\right)$
    Clip y_i to [0,1]
  end for

end while

and (48), updating both the Lagrange multipliers $\boldsymbol{\alpha}_{\hat{i}}$ and the local copies $\boldsymbol{y}_{(L,\hat{i})}$ together for each subproblem, because they are local operations that do not depend on other variables once $\boldsymbol{y}$ is updated in the previous iteration. This reveals another advantage of our inference algorithm: it is very easy to parallelize. The updates (47) and (48) can be performed in parallel, the results gathered, update (49) performed, and the updated $\boldsymbol{y}$ broadcast back to the subproblems. 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 $\hat{\boldsymbol{\phi}}$ of the index set $\{1, \ldots, m\}$ of the potentials $\boldsymbol{\phi}$. Observe that if a feasible assignment to $\boldsymbol{y}$ minimizes

$$\sum_{j \in \hat{\boldsymbol{\phi}}} w_j \phi_j(\boldsymbol{y}, \boldsymbol{x}) \tag{58}$$

and $\phi_j(\boldsymbol{y}, \boldsymbol{x}) = 0, \forall j \notin \hat{\boldsymbol{\phi}}$, 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, comparing its running times against those of MOSEK[4] a commercially available convex

[4] http://www.mosek.com

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 structured-prediction 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.

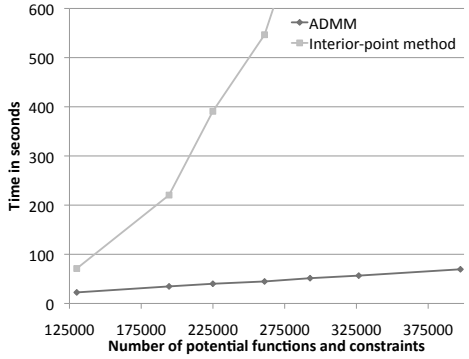
We evaluate scalability by generating social networks of varying sizes, constructing HL-MRFs with them, and measuring the running time required to find a MAP state. We compare our algorithm to MOSEK's IPM. The social networks we generate are designed to be representative of common social-network analysis 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

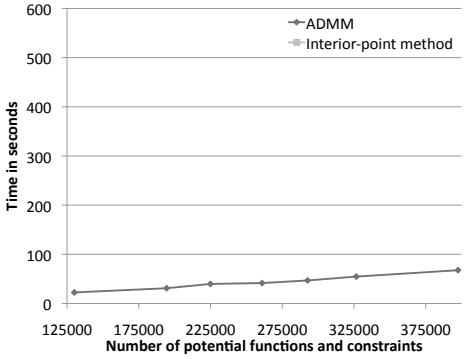
We generate social networks with between 22,050 and 66,150 vertices, which induce HL-MRFs with between 130,080 and 397,488 total potential functions and constraints. In all the HL-MRFs, between 83% and 85% of those totals are potential functions and between 15% and 17% are constraints. For each social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

We implement ADMM in Java and compare with the IPM in MOSEK (version 6) 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 $p = 0.0000005$.

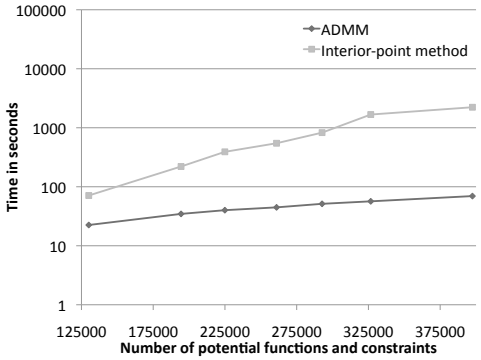
We first evaluate the scalability of ADMM when solving piecewise-linear MAP problems and compare with MOSEK's interior-point method. Figures 1a (normal scale) and 1c (log scale) show the results. The running time of the interior-point method quickly explodes as the problem size increases. The IPM's average 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



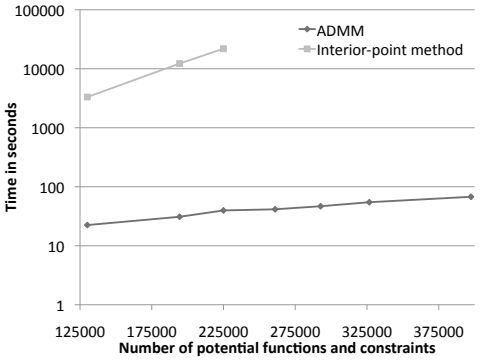
(a) Linear MAP problems



(b) Quadratic MAP problems



(c) Linear MAP problems (log scale)



(d) Quadratic MAP problems (log scale)

Figure 1: Average running times to find a MAP state for HL-MRFs.

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 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 1b (normal scale) and 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 (45) 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.

6 Weight Learning

In this section 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 $\mathcal{T} = (t_1, \ldots, t_s)$ denote a vector of templates with associated weights $\boldsymbol{W} = (W_1, \ldots, W_s)$. 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 $\phi_j(\boldsymbol{y}, \boldsymbol{x})$ was defined by template t_q. Then, we refer to the sum of the potentials defined by a template as

$$\Phi_q(\boldsymbol{y}, \boldsymbol{x}) = \sum_{j \in t_q} \phi_j(\boldsymbol{y}, \boldsymbol{x}) \ . \tag{59}$$

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

$$f_{\boldsymbol{w}}(\boldsymbol{y}, \boldsymbol{x}) = \boldsymbol{W}^\top \boldsymbol{\Phi}(\boldsymbol{y}, \boldsymbol{x}) \ , \tag{60}$$

where $\boldsymbol{\Phi}(\boldsymbol{y}, \boldsymbol{x}) = (\Phi_1(\boldsymbol{y}, \boldsymbol{x}), \ldots, \Phi_s(\boldsymbol{y}, \boldsymbol{x}))$. 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 $\boldsymbol{W}$ is to maximize the log-likelihood of training data. The partial derivative of the log-likelihood with respect to a parameter W_q is

$$\frac{\partial \log P(\boldsymbol{y}|\boldsymbol{x})}{\partial W_q} = \mathbb{E}_{\boldsymbol{W}} \left[\Phi_q(\boldsymbol{y}, \boldsymbol{x})\right] - \Phi_q(\boldsymbol{y}, \boldsymbol{x}), \tag{61}$$

where $\mathbb{E}_{\boldsymbol{W}}$ is the expectation under the distribution defined by $\boldsymbol{W}$. The voted perceptron algorithm (Collins, 2002) optimizes $\boldsymbol{W}$ 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 $\boldsymbol{y}$ 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.,

$$P^*(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{n} P^*(y_i|\mathrm{MB}(y_i), \boldsymbol{x}) \tag{62}$$

$$= \prod_{i=1}^{n} \frac{1}{Z_i(\boldsymbol{W}, \boldsymbol{y}, \boldsymbol{x})} \exp\left[-f^i_{\boldsymbol{w}}(y_i, \boldsymbol{y}, \boldsymbol{x})\right]; \tag{63}$$

$$Z(w, y_i) = \int_{y_i} \exp\left[-f^i_{\boldsymbol{w}}(y_i, \boldsymbol{y}, \boldsymbol{x})\right]; \tag{64}$$

$$f^i_{\boldsymbol{w}}(y_i, \boldsymbol{y}, \boldsymbol{x}) = \sum_{j: i \in \phi_j} w_j \phi_j \left(\{y_i \cup \boldsymbol{y}_{\setminus i}\}, \boldsymbol{x}\right). \tag{65}$$

Here, $i \in \phi_j$ means that y_i is involved in ϕ_j, and $\mathrm{MB}(y_i)$ 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

$$\frac{\partial \log P^*(\boldsymbol{y}|\boldsymbol{x})}{\partial W_q} = \sum_{i=1}^{n} \mathbb{E}_{y_i|\text{MB}} \left[\sum_{j \in t_q : i \in \phi_j} \phi_j(\boldsymbol{y}, \boldsymbol{x}) \right] - \Phi_q(\boldsymbol{y}, \boldsymbol{x}) \ . \tag{66}$$

Computing the pseudolikelihood gradient does not require inference and takes time linear in the size of $\boldsymbol{y}$. However, the integral in the above expectation does not readily admit a closed-form antiderivative, so we approximate the expectation. When a variable in 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 $\boldsymbol{W}$ 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 $\tilde{\boldsymbol{y}}$, a large-margin solution should satisfy:

$$f_{\boldsymbol{w}}(\boldsymbol{y}, \boldsymbol{x}) \leq f_{\boldsymbol{w}}(\tilde{\boldsymbol{y}}, \boldsymbol{x}) - L(\boldsymbol{y}, \tilde{\boldsymbol{y}}), \quad \forall \tilde{\boldsymbol{y}}, \tag{67}$$

where the loss function $L(\boldsymbol{y}, \tilde{\boldsymbol{y}})$ measures the disagreement between a state $\tilde{\boldsymbol{y}}$ and the training label state $\boldsymbol{y}$. A common assumption is that the loss function decomposes over the prediction components, i.e., $L(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \sum_i L(y_i, \tilde{y}_i)$. In this work, we use the ℓ_1 distance as the loss function, so $L(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \sum_i \|y_i - \tilde{y}_i\|_1$. 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

$$\begin{aligned} \min_{\boldsymbol{W} \geq 0} \quad & \frac{1}{2}||\boldsymbol{W}||^2 + C\xi \\ \text{s.t.} \quad & \boldsymbol{W}^\top (\Phi(\boldsymbol{y}, \boldsymbol{x}) - \Phi(\tilde{\boldsymbol{y}}, \boldsymbol{x})) \leq -L(\boldsymbol{y}, \tilde{\boldsymbol{y}}) + \xi, \quad \forall \tilde{\boldsymbol{y}}, \end{aligned} \tag{68}$$

where $\Phi(\boldsymbol{y}, \boldsymbol{x}) = (\Phi_1(\boldsymbol{y}, \boldsymbol{x}), \ldots, \Phi_s(\boldsymbol{y}, \boldsymbol{x}))$ 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 constrain given by a *separation oracle*, then updating $\boldsymbol{W}$ 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

$$\arg\min_{\tilde{\boldsymbol{y}}} \boldsymbol{W}^\top \Phi(\tilde{\boldsymbol{y}}, \boldsymbol{x}) - L(\boldsymbol{y}, \tilde{\boldsymbol{y}}). \tag{69}$$

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

$$\begin{aligned} \min_{\boldsymbol{W} \geq 0} \quad & \frac{1}{2}||\boldsymbol{W}||^2 + C\xi \\ \text{s.t.} \quad & K. \end{aligned} \tag{70}$$

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 Learning

To demonstrate the flexibility and effectiveness of 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 structured-prediction 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.

Table 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

For each category `"C_i"`, we have two separate rules for each direction of citation:

```
Category(A, "C_i") && Cites(A, B) -> Category(B, "C_i")
Category(A, "C_i") && Cites(B, A) -> Category(B, "C_i")
```

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

```
Category(D, +C) = 1.0 .
```

Table 1 lists the results of this experiment. HL-MRFs are the most accurate predictors on both data sets. Both variants of HL-MRFs are also much faster than discrete MRFs. See Table 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, orginally 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 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.

Table 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**

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

```
Trusts(A,B) && Trusts(B,C) -> 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 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 3 lists average inference times

Table 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

on five folds of three prediction tasks: Cora, Citeseer, and Epinions. 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 $\boldsymbol{x}$; the remaining ratings are treated as the variables $\boldsymbol{y}$.

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

```
SimRating(J1, J2) && Likes(U, J1) -> 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

Table 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**

`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 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.

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

Table 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

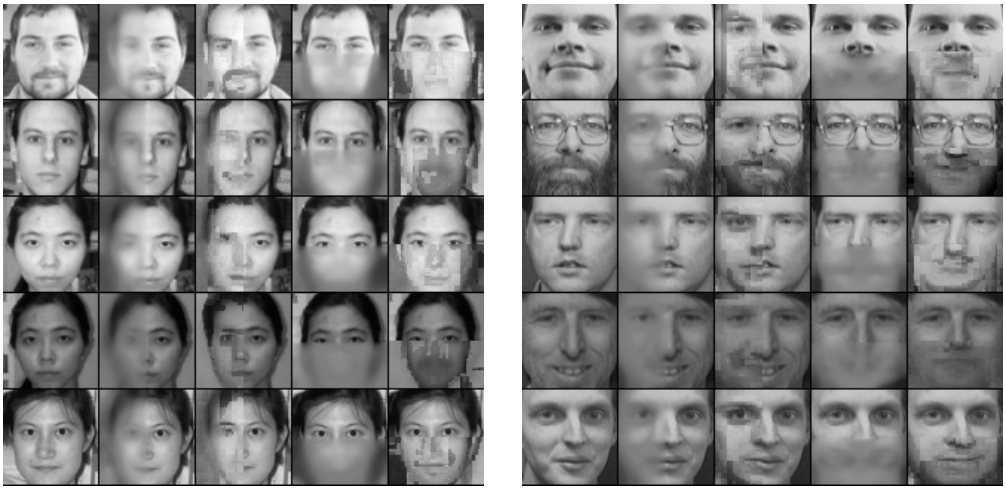


Figure 2: Example results on image completion of Caltech101 (left) and Olivetti (right) faces. From left to right in each column: (1) true face, left side predictions by (2) HL-MRFs and (3) SPNs, and bottom half predictions by (4) HL-MRFs and (5) SPNs. SPN completions are downloaded from Poon and Domingos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

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 `Bright("P_ij", I)` is the normalized brightness of pixel `"P_ij"` in image `I`, and `North("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 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 Figure2, 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.

7 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 interdependent 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 paper 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 7.1), inference methods (Section 7.2), and learning methods (Section 7.3).

7.1 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 in 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 2 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.

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 paper 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).

7.2 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., 2012).

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 AD^3 (Martins et al., 2015), which uses the alternating direction method of multipliers (ADMM). AD^3 optimizes objective (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 AD^3, 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 nonlinear 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 paper 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 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.

7.3 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 unfavored 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.

8 Conclusion

In this paper we 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 the library of machine learning tools because they embody a very useful point in the spectrum of models that trade off between scalability and expressivity. As we 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.

We also introduced PSL, a probabilistic programming language for HL-MRFs. PSL makes HL-MRFs easy to design, allowing users to encode their ideas for structural dependencies using an intuitive syntax based on first-order logic. 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, we 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 dependencies in structured prediction. The supervised learning algorithms extend standard learning objectives to HL-MRFs. 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 paper 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. With these modeling tools, researchers can design and apply new solutions to structured prediction problems.

Acknowledgements

We would like to acknowledge the many people whose advice and suggestions have contributed to the development of HL-MRFs and PSL. Contributors include Shobeir Fakhraei, James Foulds, Angelika Kimmig, Stanley Kok, Ben London, Hui Miao, Lilyana Mihalkova, Dianne P. O'Leary, Jay Pujara, Arti Ramesh, Theodoros Rekatsinas, and V.S. Subrahmanian. This work was 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 authors 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.

A Proof of Theorem 2

In this appendix, we prove the equivalence of objectives (7) and (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 (10) as

$$\operatorname*{arg\,max}_{\boldsymbol{\mu} \in [0,1]^n} \sum_{C_j \in \boldsymbol{C}} \hat{\phi}_j(\boldsymbol{\mu}), \tag{71}$$

where

$$\hat{\phi}_j(\boldsymbol{\mu}) = \max_{\boldsymbol{\theta}_j} \; w_j \sum_{\boldsymbol{x}_j | \boldsymbol{x}_j \neq \boldsymbol{x}_j^F} \theta_j(\boldsymbol{x}_j) \tag{72}$$

$$\text{such that} \sum_{\boldsymbol{x}_j | x_j(i)=1} \theta_j(\boldsymbol{x}_j) = \mu_i \qquad \forall i \in I_j^+ \tag{73}$$

$$\sum_{\boldsymbol{x}_j | x_j(i)=0} \theta_j(\boldsymbol{x}_j) = 1 - \mu_i \qquad \forall i \in I_j^- \tag{74}$$

$$\sum_{\boldsymbol{x}_j} \theta_j(\boldsymbol{x}_j) = 1 \tag{75}$$

$$\theta_j(\boldsymbol{x}_j) \geq 0 \qquad \forall \boldsymbol{x}_j \; . \tag{76}$$

It is straightforward to verify that objectives (10) and (71) 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 $\boldsymbol{\theta}_j^\star$ of $\hat{\phi}_j(\boldsymbol{\mu})$, which is guaranteed to exist because problem (72) is bounded and feasible[5] 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 $w_j \phi_j(\boldsymbol{x})$. Observe that, by the objective (72) and the simplex constraint (75), showing that $\hat{\phi}_j(\boldsymbol{\mu})$ is not fully satisfiable is equivalent to showing that $\theta_j^\star(\boldsymbol{x}_j^F) > 0$.

Lemma 16 *If*

$$\sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i) < 1 \; ,$$

then $\theta_j^\star(\boldsymbol{x}_j^F) > 0$.

Proof By the simplex constraint (75),

$$\sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i) < \sum_{\boldsymbol{x}_j} \theta_j^\star(\boldsymbol{x}_j) \; .$$

Also, by summing all the constraints (73) and (74),

$$\sum_{\boldsymbol{x}_j | \boldsymbol{x}_j \neq \boldsymbol{x}_j^F} \theta_j^\star(\boldsymbol{x}_j) \leq \sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i) \; ,$$

because all the components of $\boldsymbol{\theta}^\star$ are nonnegative, and—except for $\theta_j^\star(\boldsymbol{x}_j^F)$—they all appear at least once in constraints (73) and (74). These bounds imply

$$\sum_{\boldsymbol{x}_j | \boldsymbol{x}_j \neq \boldsymbol{x}_j^F} \theta_j^\star(\boldsymbol{x}_j) < \sum_{\boldsymbol{x}_j} \theta_j^\star(\boldsymbol{x}_j) \; ,$$

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

which means $\theta_j^\star(\boldsymbol{x}_j^F) > 0$, 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 $\theta_j^\star(\boldsymbol{x}_j^F) > 0$, then*

$$\sum_{\boldsymbol{x}_j | \boldsymbol{x}_j \neq \boldsymbol{x}_j^F} \theta_j^\star(\boldsymbol{x}_j) = \sum_{i \in I_j^+} \mu_i + \sum_{i \in I_j^-} (1 - \mu_i) \ .$$

Proof We prove the lemma via the Karush-Kuhn-Tucker (KKT) conditions (Karush, 1939; Kuhn and Tucker, 1951). Since problem (72) is a maximization of a linear function subject to linear constraints, the KKT conditions are necessary and sufficient for any optimum $\boldsymbol{\theta}_j^\star$.

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

$$d(\boldsymbol{x}_j) \triangleq \left\{ i \in I_j^+ \cup I_j^- | \boldsymbol{x}_j(i) \neq \boldsymbol{x}_j^F(i) \right\}$$

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

We now write the relevant KKT conditions for $\boldsymbol{\theta}_j^\star$. Let $\boldsymbol{\lambda}, \boldsymbol{\alpha}$ be real-valued vectors where $|\boldsymbol{\lambda}| = |I_j^+| + |I_j^-| + 1$ and $|\boldsymbol{\alpha}| = |\boldsymbol{\theta}_j|$. Let each λ_i correspond to a constraint (73) or (74) for $i \in I_j^+ \cup I_j^-$, and let λ_Δ correspond to the simplex constraint (75). Also, let each $\alpha_{\boldsymbol{x}_j}$ correspond to a constraint (76) for each $\boldsymbol{x}_j$. Then, the following KKT conditions hold:

$$\alpha_{\boldsymbol{x}_j} \geq 0 \qquad \forall \boldsymbol{x}_j \tag{77}$$

$$\alpha_{\boldsymbol{x}_j} \theta_j^\star(\boldsymbol{x}_j) = 0 \qquad \forall \boldsymbol{x}_j \tag{78}$$

$$\lambda_\Delta + \alpha_{\boldsymbol{x}_j^F} = 0 \tag{79}$$

$$w_j + \sum_{i \in d(\boldsymbol{x}_j)} \lambda_i + \lambda_\Delta + \alpha_{\boldsymbol{x}_j} = 0 \qquad \forall \boldsymbol{x}_j \neq \boldsymbol{x}_j^F \ . \tag{80}$$

Since $\theta_j^\star(\boldsymbol{x}_j^F) > 0$, by condition (78), $\alpha_{\boldsymbol{x}_j^F} = 0$. By condition (79), 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 $\boldsymbol{x}_j$ such that $d(\boldsymbol{x}_j) = \{i\}$. Then, it follows from condition (80) that there exists $\boldsymbol{x}_j$ such that, for every $i \in I_j^+ \cup I_j^-$,

$$w_j + \lambda_i + \lambda_\Delta + \alpha_{\boldsymbol{x}_j} = 0 \ .$$

Since $\alpha_{\boldsymbol{x}_j} \geq 0$ by condition (77) and $\lambda_\Delta = 0$, it follows that $\lambda_i \leq -w_j$. With these bounds, we show that, for any state $\boldsymbol{x}_j$, if $|d(\boldsymbol{x}_j)| \geq 2$, then $\theta_j^\star(\boldsymbol{x}_j) = 0$. Assume that for some state $\boldsymbol{x}_j$, $|d(\boldsymbol{x}_j)| \geq 2$. By condition (80) and the derived constraints on $\boldsymbol{\lambda}$,

$$\alpha_{\boldsymbol{x}_j} \geq (|d(\boldsymbol{x}_j)| - 1) w_j > 0 \ .$$

With condition (78), $\theta_j^\star(\boldsymbol{x}_j) = 0$. Next, observe that for all $i \in I_j^+$ (resp. $i \in I_j^-$) and for any state $\boldsymbol{x}_j$, if $d(\boldsymbol{x}_j) = \{i\}$, then $x_j(i) = 1$ (resp. $x_j(i) = 0$), and for any other state $\boldsymbol{x}_j'$

such that $x'_j(i) = 1$ (resp. $x'_j(i) = 0$), $d(\boldsymbol{x}'_j) \geq 2$. By constraint (73) (resp. constraint (74)), $\theta^\star(\boldsymbol{x}_j) = \mu_i$ (resp. $\theta^\star(\boldsymbol{x}_j) = 1 - \mu_i$).

We have shown that if $\theta^\star_j(\boldsymbol{x}^F_j) > 0$, then for all states $\boldsymbol{x}_j$, if $d(\boldsymbol{x}_j) = \{i\}$ and $i \in I^+_j$ (resp. $i \in I^-_j$), then $\theta^\star_j(\boldsymbol{x}_j) = \mu_i$ (resp. $\theta^\star_j(\boldsymbol{x}_j) = 1 - \mu_i$), and if $|d(\boldsymbol{x}_j)| \geq 2$, then $\theta^\star_j(\boldsymbol{x}_j) = 0$. This completes the proof. ■

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*

$$\sum_{i \in I^+_j} \mu_i + \sum_{i \in I^-_j} (1 - \mu_i) \geq 1 \ ,$$

then $\theta^\star_j(\boldsymbol{x}^F_j) = 0$.

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, $\theta^\star_j(\boldsymbol{x}^F_j) > 0$. Then, by Lemma 17,

$$\sum_{\boldsymbol{x}_j | \boldsymbol{x}_j \neq \boldsymbol{x}^F_j} \theta^\star_j(\boldsymbol{x}_j) \geq 1 \ .$$

The assumption that $\theta^\star_j(\boldsymbol{x}^F_j) > 0$ implies

$$\sum_{\boldsymbol{x}_j} \theta^\star_j(\boldsymbol{x}_j) > 1,$$

which is a contradiction, since it violates the simplex constraint (75). The possibility that $\theta^\star_j(\boldsymbol{x}^F_j) < 0$ is excluded by the nonnegativity constraints (76). ■

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

Lemma 19 *If $\theta^\star_j(\boldsymbol{x}^F_j) = 0$, then*

$$\sum_{\boldsymbol{x}_j | \boldsymbol{x}_j \neq \boldsymbol{x}^F_j} \theta^\star_j(\boldsymbol{x}_j) = 1 \ .$$

Proof The lemma follows from the simplex constraint (75). ■

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 $\boldsymbol{\mu}$,*

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

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,

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

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,

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

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 $\boldsymbol{\mu}^\star$ of objective (10) is an optimum $\hat{\boldsymbol{y}}^\star$ of objective (7), and vice versa.*

Proof Substituting the solution of the inner optimization from Lemma 20 into the local consistency relaxation objective (71) gives a projected optimization over only $\boldsymbol{\mu}$ which is identical to the MAX SAT relaxation objective (7). ■

References

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, and L. Getoor. Unifying local consistency and MAX SAT relaxations for scalable inference with rounding guarantees. In *Artificial Intelligence and Statistics (AISTATS)*, 2015.

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 (SocialCom)*, 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.

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.

L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3:679–707, 2002.

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.

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.

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. Jojic, 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.

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. AD^3: 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, 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 Aritificial 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.

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. 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. Globally convergent dual MAP LP relaxation solvers using Fenchel-Young margins. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

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 Aritificial 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 Aritificial 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. Tsochantaridis, 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.

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.

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.