Robust and scalable weight learning for undirected relational models

Daniel Lowd

<lowd@cs.washington.edu>

March 25, 2007

Abstract

Undirected relational models such as Markov logic networks and recursive random fields have the power and flexibility to represent many complex relational concepts. Weight learning for these models is somewhat weak in comparison. In this paper, we discuss methods that might lead to more efficient algorithms for learning with large datasets, complex models, and many-layered models. We take inspiration from conditional random fields, probabilistic models of the brain, and connectionist learning to suggest several possible ways to learn these models better.

1 Introduction

Statistical relational learning is a field of growing interest, since it brings the power of statistical learning to rich and complex domains. Of particular interest are undirected models, since they can better represent the symmetric relationships that naturally arise in relational data.

An early undirected representation was relational Markov networks (RMNs) [36]. More recently, Richardson and Domingos introduced Markov logic networks [30] which have risen in popularity due to their ability to represent any propositional Markov random field as well as any knowledge base in finite, first-order logic. Markov logic networks also generalize RMNs, conditional random fields [17], and many other statistical relational representations. Any improvement to MLN learning or inference therefore has the potential for high impact, since the framework subsumes most earlier models and continues to be applied to more and more problems

One undirected model not subsumed by Markov random fields is recursive random fields (RRFs) [20]. An RRF can be seen as a recursive generalization of an MLN that models probability at many levels. This allows RRFs to represent compactly many concepts that in an MLN would require exponential space. The cost of this greater flexibility is that RRFs are less computationally efficient on most problems and weight learning may get stuck in local optima.

The remainder of this paper, we will focus primarily on MLNs and RRFs, since they comprise the state-of-the-art in undirected relational models.

1.1 Markov Logic

A Markov logic network (MLN) is a statistical relational model consisting of real-valued weights and formulas in first-order logic, $\{(w_i, f_i)\}$ [30]. When grounded, these formulas become the features of a Markov random field, where each node is a ground predicate. This leads to the following joint probability distribution:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right)$$

Where Z is the constant required to normalize the probability distribution. Note that computing the partition function, Z, is generally intractable, making most probabilistic computations intractable as well.

MAP inference can be approximated using the MaxWalkSAT algorithm [14]. General probabilistic inference can be done by the MC-SAT algorithm [26], an MCMC slice sampler that is much better at transitioning between modes than simple Gibbs sampling.

For learning, we rarely need to model the joint distribution of all predicates, and therefore seek to optimize the conditional log likelihood (CLL). The gradient of the CLL for an MLN with respect to the weights is the difference between the actual and expected counts of each feature:

$$\frac{\partial}{\partial w_i} \log P(Y = y|X = x) = n_i(y) - E_X[n_i(y)]$$

The current state-of-the-art method for discriminative weight learning in MLNs is the voted perceptron algorithm with an MPE approximation, as proposed by Collins [6]. This is a basic steepest-descent weight optimization procedure, in which the expected counts are approximated using MaxWalkSAT and the weights are averaged across all learning iterations. The voted perceptron algorithm has also been applied to relational Markov networks by Bunescu and Mooney [4].

Alternatively, one can optimize pseudo-likelihood [3], which conditions each random variable in the model on its Markov blanket:

$$P^*(X{=}x) = \prod_{t=1}^{n} P(X_t = x_t)$$

The advantage of pseudo-likelihood is that it can be computed exactly and efficiently. Richardson and Domingos [30] optimized pseudo-likelihood using the limited memory BFGS (L-BFGS) quasi-Newton method for this very reason. The disadvantage of pseudo-likelihood is that it tends to do poorly with long chains of inference.

1.2 Recursive Random Fields

One way to think about an MLN is as a probabilistic softening of the top-level conjunction and universal quantifiers in a deterministic knowledge base. Recursive random fieleds (RRFs) take this a step further by softening connectives and quantifiers at every level in a knowledge base.

This is accomplished by recursively replacing the formulas in an MLN with log-linear models. A recursive random field can be viewed as a directed acyclic graph of parameterized features, or "parfeatures." Each parfeature is either a ground predicate or a log-linear model of other parfeatures:

$$\begin{aligned} f_{i,\vec{g}}(\mathbf{x}) &= R_i(g_{i_1}, \ldots, g_{i_k}) \quad \text{(base case)} \\ f_{i,\vec{g}}(\mathbf{x}) &= \frac{1}{Z_i} \exp\left(\sum_j w_{ij} \sum_{\vec{g'}} f_{j,\vec{g},\vec{g'}}(\mathbf{x}) \right) \quad \text{(recursive case)} \end{aligned}$$

Where the summation is over the children of f_i according to the graph, and the vector subscripts $\vec{g}$ and $\vec{g'}$ represent the parfeature parameters, objects in the domain used for grounding.

The full joint probability distribution is simply given by the root feature, f_0:

$$P(X{=}x) = f_0(\mathbf{x})$$

Although we specify a normalization constant Z_i for each feature f_i, all features except for the root are multiplied by some weight. Therefore, we can fold this normalization into the weight and ignore it, or use whatever normalization is convenient for learning. The partition function at the top level is still required to ensure that we have a valid probability distribution.

With a sufficiently general structure, a two-level RRF can represent any MLN of a certain complexity, just as two-level neural networks can compute any logical function given enough hidden nodes. In addition, RRFs can compactly represent many probability distributions that an MLN cannot efficiently represent. This includes m-of-n concepts that neural networks can represent effciently but Boolean logic cannot. Since RRFs are relational, they can also represent m-of-all concepts, such as, “Everyone has at least k friends." Here, the effective value of k depends on the weights.

In fact, just by adjusting the weights, a single RRF structure can represent many different MLN structures (as well as many distributions an MLN cannot compactly model). Therefore, if our weight learning algorithm is good enough, then we should be able to effectively learn structure by learning weights.

Thus far, limited experiments have been done with contrastive divergence approximately optimizing conditional log-likelihood [19, 11] and LBGFS optimizing pseudo-likelihood [20, 3]. These experiments confirm that RRFs with the right structure can outperform MLNs when an m-of-all concept is present.

1.3 The Challenge of Weight Learning

Weight learning in MLNs and RRFs is difficult for several reasons. First, as noted earlier, computing the likelihood is intractable. This means that any optimization algorithm will depend heavily on approximate inference. The rate of convergence and quality of solution therefore depend on the bias and variance of the inference mechanism.

Furthermore, the domains are often very large. The most canonical, large, relational dataset is the world wide web, but even relatively modest relational domains become large through their representation. In a relational domain, the number of ground predicates is polynomial in the number of objects. The number of ground clauses or parfeatures is typically a higher order polynomial. Given a social network domain with a $Friends(A, B)$ predicate, a social network of 1000 people would have 1 million groundings of the Friends predicate and 1 trillion groundings of the transitivity rule,

$Friends(A, B) \wedge Friends(B, C) \Rightarrow Friends(A, C)$.

Many relational domains also have a large number of attributes or relations, leading to a large number of weighted rules. For example, many language-based domains have word-specific rules. The Alchemy system [16] even has syntactic sugar to better support these kinds of rules.

The size and complexity of these domains makes weight learning very difficult. Consider the experiments of Singla and Domingos on the Cora dataset [33]. Their task was to determine whether or not two bibliographic citations refer to the same research paper. After restricting the query predicates to likely matches using canopies, they still had over 60,000 match decisions. Without the canopies, learning would be completely intractable. Even with the canopies, learning took over 2 gigabytes of memory. Furthermore, they achieved the best results with per-word weights, but were unable to learn these weights from the data. Instead, Singla and Domingos had to set weights using a naive Bayes approximation. Datasets like Cora are not uncommon.

In addition, these learning problems tend to be highly ill-conditioned, making them very difficult to solve by steepest-descent methods. An optimization problem is said to be "ill-conditioned" when the condition number of the Hessian matrix is large, where the condition number is the ratio of its largest to smallest Eigenvalues. This indicates a highly skewed valley through which gradient descent is likely to slowly zig-zag.

The reason why MLNs tend to be ill-conditioned is visible in the form of the Hessian and the nature of relational features. The Hessian matrix for an MLN is simply the negated covariance matrix of feature counts. For example, each diagonal entry in the Hessian is the variance of one MLN feature. Since some MLN features may have many more groundings than other MLN features, this often leads to widely different variances and a very ill-conditioned Hessian. In practice, Lowd and Domingos (personal communication) found an MLN for the Cora dataset to have an estimated condition number of over 600,000 at the start of training when all weights were zero.

RRFs inherit all of the challenges of MLNs along with several new problems. First of all, inference tends to be slower because features can have many different values, while MLN features are either true or false. That is, MLN inference algorithms can often short-circuit computation when a formula is known to be satisfied or unsatisfied; in an RRF, all sub-features must be known to compute the feature value.

Furthermore, the multi-layered structure of RRFs leads to many local optima, whereas weight learning in MLNs is a convex optimization problem. This means that the initial choice of weights is very important for RRFs: relatively small changes in the weights can have a large effect on the quality of the model learned. Equally important is the choice of structure, since some structures may be more amenable to learning than others.

2 Weight Learning in CRFs

Both Markov logic networks and conditional random fields (CRFs) [17] are Markov random fields when instantiated, so most learning procedures used on CRFs are also applicable to MLNs, and may even be applicable to recursive random fields. What makes MLNs and RRFs harder to learn than CRFs is their relational features, which lead to highly varied numbers of groundings and very ill-conditioned learning problems.

In contrast, CRF features tend to be properties of adjacent objects in a linear chain, leading to each having a similar number of groundings. In this section, we review the state-of-the-art techniques for weight learning in CRFs and discuss how they might be applied or adapted to more general-purpose relational weight learning.

CRFs were first trained using an iterative-scaling type algorithm [17]. Wallach [39] found conjugate gradient and quasi-Newton methods to be substantially more efficient. Sha and Pereira [32] found the L-BFGS quasi-Newton method and preconditioned conjugate gradient to be the most efficient and effective methods for training a CRF.

All of those results are on linear chain CRFs, for which inference is tractable. Many CRFs have loops, often induced by global constraints, making inference intractable. One example is dynamic conditional random fields, which McCallum et al. were able to learn using belief propagation and L-BFGS [21].

2.1 Approximating the Log Likelihood

If the log likelihood of an MLN or RRF could be efficiently and robustly approximated, then we could safely use line search methods such as L-BFGS and conjugate gradient. These methods potentially offer much faster convergence than the voted perceptron algorithm, which can be seen as an approximate version of simple gradient descent.

One approach is to compute the marginal probability of each query atom in the training data. This could be done via sampling (e.g., MC-SAT [26]), belief propagation [23], or variational inference using the mean field approximation [13]. Most probability estimation in MLNs is currently done via sampling, but the sampling noise could confuse a line search, leading to an unstable learning algorithm.

Belief propagation is likely to work better. In fact, belief propagation has been used with conjugate gradient to learn RMNs [36] and with L-BFGS to learn CRFs [21]. The biggest downside of belief propagation is that it may not converge. Taskar et al. [36] had some trouble with belief propagation not converging on RMNs and attributed it to an especially loopy linking structure over the web pages being modeled. Since MLNs offer more flexible feature construction than RMNs, MLN graphs could potentially be more loopy and have even more trouble with inference. This is also true of RRFs.

In such cases, variational inference could be the best bet, since it is guaranteed to converge and provides a lower bound on the likelihood. The most basic approach is the mean field approximation. If the graph has a regular structure, then there may be an efficient variational approximation that does better.

Note that these methods compute marginal probabilities, which could systematically underestimate the joint likelihood even if the marginal probabilities are correct. This is especially the case when there are multiple modes. Consider a two-variable dataset in which $P(A) = 0.5$, $P(B) = 0.5$, and $P(A = B) = 1$. The marginal probabilities alone cannot convey that A and B are highly correlated. However, if the MLN or RRF has enough evidence, then the conditional distribution could have a single mode even if the joint distribution has many modes. In the two-variable example, if A is evidence, then the conditional distribution $P(B|A)$ has a single mode.

Variational inference can also be used to directly approximate log Z, from which we can compute the log likelihood directly via the clause counts. A promising approach by Wainwright et al. [38] uses spanning trees to find an upper bound for Z. Sutton and McCallum [34] adopt a much simpler approach: they approximate Z by breaking the graph up into disjoint edges. This leads to a piecewise training method, in which the parameters on each edge can be optimized independently using exact inference and L-BFGS. This method is somewhat reminiscent of pseudo-likelihood, except that instead of assuming that most of the graph is evidence, they assume that most of the graph is non-existent. They report better results than pseudo-likelihood based training for an information extraction domain. Even more impressive, they report better results than belief propagation based training, which only converged when carefully initialized. This result suggests that variational approaches may be more appropriate than belief propagation for undirected relational models. Another promising result of the piecewise training method is that it has already been applied to a Markov logic network by Culotta and McCallum [7].

The problem with piecewise training is that it assumes only pairwise potentials, while most MLNs and RRFs have potential functions of higher and lower arity. Even simple prior probabilities on the predicates, as are often represented by unit clauses in MLNs, are not handled by piecewise training. To address this, Sutton and Minka [35] have developed several generalizations of piecewise training that correspond to running one or two parallel iterations of belief propagation. These generalizations can handle unit priors as well as hyper-edges. They achieve promising results at estimating the gradient on synthetic data. How well these approximations work on real problems remains to be seen, but Markov logic could be an excellent test bed.

2.2 Avoiding the Log Likelihood

A second approach to applying second-order optimization methods is to avoid approximating the likelihood altogether and simply use the information in the approximate gradient and Hessian matrix.

Second-order methods typically choose a direction and then optimize along that direction. This optimization is either done as an iterative line search or using a "trust region." Trust region methods use the Hessian or its inverse to find the optimal step with length less than a certain threshold, defining a region in which the second-order approximation can be trusted to be good. The size of the trust region is updated heuristically by comparing the size of the predicted change in function value with its actual change – when the approximation is good, the trust region grows; when it is poor, the trust region shrinks. See Nocedal and Wright [25] for background and additional details.

Trust region methods still use function evaluations to evaluate if a step is successful and update the size of the trust region. In general, computing the log likelihood of an MLN or RRF is very expensive. Instead of approximating the likelihood, we may be able to avoid it altogether by adapting trust region methods to use the change in the gradient of the function rather than the function value itself[1]. Such methods may not behave as well as traditional trust region methods, but at least for MLNs the convexity of the optimization problem should make them stable.

While line search methods are more common, most optimization methods can be adapted to a trust region framework. One example of this is the scaled conjugate

[1]This idea was partly developed through discussions with my advisor, Pedro Domingos.

gradient algorithm, a modification of the conjugate gradient algorithm that uses a trust region [22]. The L-BFGS algorithm has also been adapted to a trust region framework by Burke and Weigmann [5], who found that a similar number of function evaluations are required by both methods.

The SR1 algorithm is more commonly used within a trust region framework [25]. SR1 is a simpler algorithm than L-BFGS that is often competitive in practice, although it lacks the global convergence guarantees of L-BFGS. Since MLN weight learning should be a reasonably well-behaved function, a limited-memory version of SR1 should be effective. RRFs could prove more challenging, since they can have many local optima.

For all of these methods, if we could effectively replace function evaluations with gradient computations, then we could apply state-of-the-art second-order optimization methods to the problem of learning MLNs and RRFs. This has the potential to find better weights faster. The biggest risk of these approaches is overfitting.

2.3 Stochastic Meta-descent

Vishwanathan et al. [37] apply the stochastic meta-descent (SMD) algorithm to the task of learning weights in CRFs. SMD has two advantages cover conjugate gradient descent: it never needs to evalute the function value, since it performs no line search, and it is robust to noise. The robustness to noise is particularly important because it means that the training set can be split up into much smaller batches. Each iteration of the optimization algorithm computes the gradient using one batch of the training data, which can consist of as little as one example. The increase in noise is offset by the fact that each training iteration is much, much faster. This is likely to be most effective when the training data is highly redundant.

The algorithm works like a simple gradient descent, except that the learning rates in all dimensions are updated in each iteration. This is accomplished by performing a second gradient descent on learning rates, using second-order information in the form of Hessian vector products which can be computed efficiently without storing the full Hessian. The adjustment of the learning rates happens in log space in order to avoid negative learning rates and to range freely over the orders of magnitude.

Many MLN and RRF learning problems consist of 1-5 relational examples, preventing us from saving much time by using smaller batch sizes. The claim of being robust to noise is much more appealing, since MLN approximate inference methods are bound to be noisy.

In unpublished experiments, Lowd and Domingos (personal communication) found the SMD algorithm to be very unstable on challenging MLN learning problems. Specifically, it would take very large steps and diverge unless gains were set very low. Schraudolph notes that SMD occassionally diverges, especially when one of the tuning parameters is too high or too low [31]. We hypothesize that the degree of ill-conditioning in our MLN learning problems exacerbates any instability in the SMD algorithm.

2.4 Combining Local Classifiers

Punyakanok et al. [27, 28] propose learning local classifiers to label sequence data and applying global constraints to get better predictions. When the amount of data is limited, they find theoretically and empirically that classifiers should be trained in ignorance of the global constraints [29]. When ample data is available, however, optimizing the classifiers based on their constrained output can yield the best performance. This is somewhat reminiscent of Sutton and McCallum's piecewise training approach [34], since both suggest learning local models to handle sequence data.

Local classifiers make a great deal of sense when there are obvious features and a few global constraints. For example, we can perform entity resolution by training classifiers to decide if two objects are equivalent and later applying a transitive closure constraint. The advantage of such approaches is that they can utilize effective propositional classifiers as well as long-distance interactions. The disadvantage is that they are somewhat ad hoc, and not all relational problems are amenable to the approach.

Overall, this approach seems more useful as a knowledge engineering heuristic than as a generic algorithm. The output of an external classifier could be a very useful input to a relational learning system, especially when the classifier is too complex or the data is too limited to learn everything in an integrated model. When enough data is available and learning the full model is tractable, one can expect to do better with an integrated model.

3 Weight Learning in the Brain

The human brain may be the most powerful and scalable learning and inference engine in existence. If we could develop computer models with structures and algorithms similar to those in the brain, perhaps we could build a truly scalable learning system. While much of how the brain works is still unknown, there has been promising research into visual perception and how it may relate to Bayesian inference.

Dean [8] calls on the work of Lee and Mumford [18] to argue the importance of better probabilistic models of the brain. Specifically, he argues that models of the brain should take the form of undirected probabilistic models in which each layer only interacts with adjacent layers. The lower levels of the model respond to the visual input, but get contextual feedback from the higher layers in the network. Through probabilistic inference methods such as belief propagation or particle filtering, they can maintain multiple hypotheses that are disambiguated by the longer-range connections of the higher level layers.

George and Hawkins [9] model visual perception as an tree-structured Bayesian network. The leaves at the bottom of the tree are the observations. At higher and higher levels, the variables model wider scale interactions among the lower-level nodes. These layers are learned one level at a time, from the bottom up. The tree structure of the Bayesian network allows for efficient inference, but it also significantly limits the complexity of the model. For example, the receptive fields at any given level do not overlap. This means that the amount of interaction between two adjacent pixels depends on their exact positioning in the visual field. If they are at the very center of the field, then they are only linked at the highest level of the network.

The important theme in this work is that longer-range interactions are handled at higher levels. This allows inference to happen locally with feedback from global context. A structure like this could lead to more localized inference and a less loopy graph, making approximate inference more accurate.

This type of hierarchy is largely absent from Markov logic networks, but it doesn't have to be. Most MLNs are constructed from knowledge bases, and therefore only represent relationships among observed variables. However, an MLN can have hidden variables represented as relations that are never observed in the training data, as discussed by Kok and Domingos[15]. The MLN can then be represented as several layers, where variables from one layer only appear in formulas with variables from adjacent layers.

Recursive random fields usually do have a hierarchy, but the nodes in the hierarchy are deterministic functions, not random variables. In other words, many variables interact with each other directly, but the function that describes their interaction is compactly represented as a graph.

4 Weight Learning in Connectionist Networks

Since recursive random fields (RRFs) can be seen as a type of neural network (and are in fact learned using back-propagation), it makes sense to look to connectionist learning for inspiration on how to learn them. These models also take inspiration from the neuroscience discussed in the previous section.

4.1 Restricted Boltzmann Machines

A Boltzmann machine [1] can be seen as a type of Markov random field with only pairwise potentials among the variables. Boltzmann machines are largely impractical, though, because parameter learning is very difficult.

Restricted Boltzmann machines (RBMs) restrict the structure of the Markov random field to a bipartite graph, so that potentials are only allowed between hidden and observed variables. Note that the hidden variables are conditionally independent given the visible variables, and vice versa. This makes it possible to perform alternating Gibbs sampling, in which each layer is sampled conditioned on the other. While this sampling could take a long time to fully converge, Hinton [11] reports good results with only a few steps of sampling. He refers to this as "contrastive divergence," since it optimizes a slightly different objective function than log likelihood.

Since an RBM is still a Markov random field, it is easy to represent any RBM as an MLN: the hidden variables simply become predicates that are never observed in the training data. To make an RBM relational, we can replace the hidden variables with hidden relations over one or more objects.

Construction of a relational RBM requires several choices, beginning with the number and arity of the hidden relations. Hidden relations over more objects can represent more complicated relationships, but will also have more groundings, slowing down learning and inference. A good approach might be to start with low arity relations and add higher arity relations once an initial model has been learned.

Second, one must select the links between the hidden and observed relations. Consider the hidden relation $H_1(A,B,C)$. A natural set of connections would be to all groundings of all predicates that only use the parameters A, B, and C. Given a binary relation R_1, such rules would include $H_1(A,B,C) \Leftrightarrow R_1(A,A)$, $H_1(A,B,C) \Leftrightarrow R_1(A,B)$, $H_1(A,B,C) \Leftrightarrow R_1(B,A)$, etc.

A hidden relation along with rules linking it to observed relations can easily represent a noisy conjunction. However, to represent an n-way noisy disjunction we must represent the n different ways that the disjunction could be satisfied via approximately n hidden relations. This increase in modeling complexity could nonetheless be worthwhile if it leads to a sufficient improvement in learning and inference. A useful analogy is the representation of propositional knowledge bases in DNF and CNF form: inference in a DNF formula can be done in linear time, but converting from CNF to DNF may result in an exponential increase in the size of the formula. Our case is less extreme: the change in efficiency is unknown, and the increase in size is at most quadratic, not exponential.

4.2 Deep Belief Networks

Deep belief networks [10] model a propositional joint probability distribution as a sigmoid belief network [24] with a special training procedure that allows it to efficiently learn with multiple layers of hidden nodes.

The trick Hinton et al. use to learn a neural network with many layers is to learn one layer at a time, assuming that there are an infinite number of layers above the current layer, all with tied weights. Furthermore, they assume that these layers create a prior distribution which renders nodes in the current layer independent, even given the values of their children. This assumption makes their variational approximation exact by removing the "explaining away" phenomenon that makes inference difficult.

It turns out that learning an infinite neural network with tied weights is equivalent to learning a restricted Boltzmann machine (RBM). In other words, a deep belief net is constructed by learning a stack of RBMs, one on top of another, fixing the weights of lower layers. Finally, a weight learning pass is performed in which all weights are adjusted simultaneously. This is slower than the initial greedy learning steps, but can help to fine-tune the model.

Hinton et al.[10] apply this model to the task of digit recognition by first training two levels of RBMs without any digit labels, and then training a final RBM level with digit labels as input. They report very good results without relying on specially constructed features. This is somewhat remarkable, since learning deep networks was long considered too difficult: back-propagation would inevitably get stuck in local optima and learn a very poor model.

The details of why this seems to work have been investigated in by Bengio et al. [2]. They find that the key to deep belief networks is the unsupervised initialization. That is, by initially training neural networks to learn the distribution of the input data, they perform dimensionality reduction to a space where discriminative learning of a different objective function is easier. It is not surprising, then, that Hinton and Salakhutdinov [12] report good results for doing pure dimensionality reduction on face data using a similar method.

We have already described how an MLN could be structured and trained as a relational RBM. This representation could conceivably be extended to multiple layers using a similar layer-by-layer training procedure. The resulting model could be thought of as a Deep And Relational Network (DARN). Whether these DARN models would work well in practice is an open question.

The success of deep belief nets also suggests that a greedy, layerwise approach might be appropriate for learning recursive random fields. However, in RRFs the different layers are computational units, not random variables. Sampling them as random variables would make little sense. This prevents us from performing alternating Gibbs sampling. Furthermore, since the nodes are summarizing the interactions of their children rather than coordinating them, they wouldn't necessarily constitute an effective dimensionality reduction. Since there is little literature on learning multi-layer models similar to RRFs, this may still be an approach worth trying.

5 Conclusion

While work on general purpose weight learning algorithms for Markov logic and recursive random fields is somewhat limited, there is plenty to be learned from work on conditional random fields, studies of the brain, and neural networks.

For learning arbitrary MLNs and RRFs with prespecified structures, the most promising approach is to apply conjugate gradient and quasi-Newton methods, either with an approximation of the likelihood or in a modified trust-region framework. The most standard approach for similar models has been to run belief propagation in conjunction to approximate the log likelihood. Since inference must be stable and these models are likely to be very loopy, variational approximations might offer better results. The approximations of Sutton and McCallum [34] and Sutton and Minka [35] look especially promising due to their efficiency and accuracy on real-world problems.

For automatically learning the structure of a domain, a greedy approach in which one layer is learned at a time looks most promising. For RRFs, each layer consists of parfeatures based on lower level parfeatures. For MLNs, we can construct relational versions of restricted Boltzmann machines and deep belief nets to learn models in which all predicate interactions take place through hidden relations. Taking inspiration from computational neuroscience, it may also be possible to achieve more efficient inference by using higher layers of the model to handle longer range interactions.

References

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[2] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. Technical Report 1282, Université de Montréal, Montreal, Canada, 2006.

[3] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24:179–195, 1975.

[4] R. Bunescu and R. J. Mooney. Collective information extraction with relational markov networks. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-2004)*, pages 439–446, Barcelona, Spain, 2004.

[5] J. Burke and A. Wiegmann. Notes on limited memory bfgs updating in a trust-region framework, 1996.

[6] M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, Philadelphia, PA, 2002.

[7] A. Culotta and A. McCallum. Practical markov logic containing first-order quantifiers with application to identity uncertainty. In *Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, pages 41–48, New York City, New York, 2006. Association for Computational Linguistics.

[8] T. Dean. A computational model of the cerebral cortex. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pages 938–943, Cambridge, MA, 2005. MIT Press.

[9] D. George and J. Hawkins. A hierarchical bayesian model of invariant pattern recognition in the visual cortex. In *Proceedings of the International Joint Conference on Neural Networks*. IEEE, 2005.

[10] G. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[11] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[12] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006.

[13] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 105–161. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[14] H. Kautz, B. Selman, and Y. Jiang. A general stochastic approach to solving problems with hard and soft constraints. In D. Du, J. Gu, and P. M. Parda-

los, editors, *The Satisfiability Problem: Theory and Applications*, pages 573–586. American Mathematical Society, New York, NY, 1996.

[15] S. Kok and P. Domingos. Towards statistical predicate invention. In *Proceedings of the ICML-2006 Workshop on Open Problems in Statistical Relational Learning*, Pittsburgh, PA, 2006. IMLS.

[16] S. Kok, P. Singla, M. Richardson, and P. Domingos. The Alchemy system for statistical relational AI. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2005. http://www.cs.washington.edu/ai/alchemy/.

[17] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, Williamstown, MA, 2001. Morgan Kaufmann.

[18] T. S. Lee and D. Mumford. Hierarchical Bayesian inference in the visual cortex. *Journal of the Optical Society of America A*, 20(7), 2003.

[19] D. Lowd and P. Domingos. Recursive random fields (workshop version). In *Proceedings of the ICML-2006 Workshop on Open Problems in Statistical Relational Learning*, Pittsburgh, PA, 2006. IMLS.

[20] D. Lowd and P. Domingos. Recursive random fields. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, Hyderabad, India, 2007. Morgan Kaufmann.

[21] A. McCallum, K. Rohanimanesh, and C. Sutton. Dynamic conditional random fields for jointly labeling multiple sequences. In *Workshop on Syntax, Semantics, Statistics; 16th Annual Conference on Neural Information Processing Systems*, 2004.

[22] M. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.

[23] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, Stockholm, Sweden, 1999.

[24] R. M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113, 1992.

[25] J. Nocedal and S. Wright. *Numerical Optimization, Second Edition*. Springer, New York, NY, 2006.

[26] H. Poon and P. Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, Boston, MA, 2006. AAAI Press.

[27] V. Punyakanok and D. Roth. The use of classifiers in sequential inference. In *Advances in Neural Information Processing Systems 13*, 2001.

[28] V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Semantic role labeling via integer linear programming inference. In *Proceedings of COLING-4*, 2004.

[29] V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Learning and inference over constrained output. In *Proceedings of the Nineteenth International Joint Conference*

on Artificial Intelligence, pages 1124–1129, Edinburgh, Scotland, 2005. Morgan Kaufmann.

[30] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

[31] N. N. Schraudolph. Local gain adaptation in stochastic gradient descent. In *Proceedings of the International Conference on Artificially Neural Networks*, pages 569–574, Edinburgh, Scotland, 1999. IEE.

[32] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics*. 2003.

[33] P. Singla and P. Domingos. Entity resolution with Markov logic. In *Proceedings of the Sixth IEEE International Conference on Data Mining*, pages 572–582, Hong Kong, 2006. IEEE Computer Society Press.

[34] C. Sutton and A. McCallum. Piecewise training for undirected models. In *Proceedings of the Twenty-first Conference on Uncertainty in Artificial Intelligence*, 2005.

[35] C. Sutton and T. Minka. Local training and belief propagation. Technical Report TR-2006-121, Microsoft Research, 2006.

[36] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 485–492, Edmonton, Canada, 2002. Morgan Kaufmann.

[37] S. Vishwanathan, N. Schraudolph, M. Schmidt, and K. Murphy. Accelerated training conditional random fields with stochastic gradient methods. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, Pittsburgh, PA, 2006.

[38] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51:2313–2335, 2005.

[39] H. Wallach. Efficient training of conditional random fields. Master's thesis, University of Edinburgh, 2002.