Recursive Random Fields

Daniel Lowd LOWD@CS.WASHINGTON.EDU
Pedro Domingos PEDROD@CS.WASHINGTON.EDU
Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195-2350, USA

Abstract

A formula in first-order logic can be viewed as a tree, with a logical connective at each node, and a knowledge base can be viewed as a tree whose root is a conjunction. Markov logic (Richardson & Domingos, 2006) makes this conjunction probabilistic, as well as the universal quantifiers directly under it, but the rest of the tree remains purely logical. This causes an asymmetry in the treatment of conjunctions and disjunctions, and of universal and existential quantifiers. We propose to overcome this by allowing the features of Markov logic networks (MLNs) to be nested MLNs. We call this representation *recursive random fields (RRFs)*. RRFs can represent many first-order distributions exponentially more compactly than MLNs. We perform inference in RRFs using MCMC and ICM, and weight learning using a form of backpropagation. Weight learning in RRFs is more powerful than structure learning in MLNs. Applied to first-order knowledge bases, it provides a very flexible form of theory revision.

1. Introduction

Recent years have seen the development of increasingly powerful combinations of relational and probabilistic representations, along with inference and learning algorithms for them. One of the most general representations to date is Markov logic, which attaches weights to first-order formulas and views them as templates for features of Markov random fields (Richardson & Domingos, 2006). While Markov logic may be the language of choice for many applications, its unification of logic and probability is incomplete. This is because it only treats the top-level conjunction and universal quantifiers in a knowledge base as probabilistic, when in principle any logical combination can be viewed as the limiting case of an underlying probability distribution. In Markov logic, disjunctions and existential quantifiers remain deterministic. Thus the symmetry between conjunctions and disjunctions, and between universal and existential quantifiers, is lost (except in the infinite-weight limit).

For example, an MLN with the formula $R(X) \wedge S(X)$ can treat worlds that violate both $R(X)$ and $S(X)$ as worse than worlds that only violate one. Since an MLN acts as a soft conjunction, the groundings of $R(X)$ and $S(X)$ simply appear as distinct formulas. (MLNs convert the knowledge base to CNF before performing learning or inference.) This is not possible for the disjunction $R(X) \vee S(X)$: no distinction is made between satisfying both $R(X)$ and $S(X)$ and satisfying just one. Since a universally quantified variable is effectively a conjunction over all groundings, while an existentially quantified variable is a disjunction over all groundings, this leads to the two quantifiers being handled differently.

This asymmetry can be avoided by "softening" disjunction and existential quantification in the same way that Markov logic softens conjunction and universal quantification. The result is a representation where MLNs can have nested MLNs as features. We call these recursive Markov logic networks, or *recursive random fields (RRFs)* for short.

RRFs have many desirable properties, including the ability to represent distributions like noisy DNF, rules with exceptions, and m-of-all quantifiers much more compactly than MLNs. RRFs also allow more flexibilty in revising first-order theories to maximize data likelihood. Standard methods for inference in Markov random fields are easily extended to RRFs, and weight learning can be carried out efficiently using a variant of the backpropagation algorithm.

RRF theory revision can be viewed as a first-order probabilistic analog of the KBANN algorithm, which initializes a neural network with a propositional theory and uses backpropagation to improve its fit to data (Towell & Shavlik, 1994). A propositional RRF (where all predicates have zero arity) differs from a multilayer perceptron in that its output is the joint probability of its inputs, not the regres-

Presented at the ICML Workshop on Open Problems in Statistical Relational Learning, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

sion of a variable on others (or, in the probabilistic version, its conditional probability). Propositional RRFs are an alternative to Boltzmann machines, with nested features playing the role of hidden variables. Because the nested features are deterministic functions of the inputs, learning does not require EM, and inference does not require marginalizing out variables.

The remainder of this paper is organized as follows. We begin by discussing MLNs and their limitations. We then introduce RRFs along with inference and learning algorithms for them, compare them to MLNs, and present preliminary experimental results.

2. Markov Logic Networks

A Markov logic network (MLN) consists of a set of formulas and weights, $\{(w_i, f_i)\}$, that serve as a template for constructing a Markov random field. Each feature of the Markov random field is a grounding of one of the formulas. The joint probability distribution is therefore:

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

where n_i is the number of true groundings of the ith formula given this assignment, and Z is a normalization constant so that the probabilities of all worlds sum to one. Richardson and Domingos (2006) show that, in finite domains, this generalizes both first-order logic and Markov random fields.

Another way to think about a Markov logic network is as a "softening" of a deterministic knowledge base. A first-order knowledge base can be represented as a conjunction of all groundings of all of its formulas. MLNs relax this conjunction by allowing worlds that violate formulas, but assign a per-grounding penalty for each formula. Worlds that violate many formulas are therefore possible, but less likely than those that violate fewer. In this way, even inconsistent knowledge bases can be applied.

However, while MLNs soften conjunctions and universals, disjunctions and existentials remain deterministic. For example, in Markov logic $\forall x\ R(x)$ (with some finite weight) assigns increasing probability to states with increasing number of objects x satisfying $R(x)$, but $\exists x\ R(x)$ assigns the same probability to all states in which at least one object satisfies $R(x)$.

3. Recursive Random Fields

A recursive random field (RRF) is a log-linear model in which each feature is either an observable variable or the output of another recursive random field. To build up intuition, we first describe the propositional case, then generalize it to the more interesting relational case.

3.1. Propositional RRFs

While our primary goal is solving relational problems, RRFs may be interesting in propositional domains as well. Propositional RRFs extend Markov random fields and Boltzmann machines in the same way multilayer perceptrons extend single-layer ones. The extension is very simple in principle, but allows RRFs to compactly represent important concepts, such as m-of-n. It also allows RRFs to learn features via weight learning, which could be more effective than current feature-search methods for Markov random fields.

The probability distribution of a propositional RRF is as follows:

$$P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z_0} \exp\left(\sum_i w_i f_i(\mathbf{x})\right)$$

where Z_0 is a normalization constant, to ensure that the probabilities of all possible states $\mathbf{x}$ sum to 1. What makes this different from a standard Markov random field is how the features $f_i(\mathbf{x})$ are defined: either

$$\begin{aligned} f_i(\mathbf{x}) &= x_j \quad \text{(base case), or} \\ f_i(\mathbf{x}) &= \frac{1}{Z_i} \exp\left(\sum_j w_{ij} f_j(\mathbf{x})\right) \quad \text{(recursive case)} \end{aligned}$$

In the recursive case, the summation is over all features f_j referenced by the "parent" feature f_i. The child features, f_j, can reference children of their own, and so on for an arbitrary number of levels. A child feature can appear in more than one parent feature, and thus an RRF can be viewed as a directed acyclic graph of features. (Note that the probabilistic graphical model it represents is still undirected.)

Since the overall distribution is simply a recursive feature, we can also write the probability distribution as follows:

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

Except for Z_0 (the normalization of the root feature, f_0), the per-feature normalization constants Z_i can be absorbed into the corresponding feature weights w_{ki} in their parent features f_k. Therefore, the user is free to choose any convenient normalization or even no normalization at all.

It is easy to show that this generalizes Markov random fields with conjunctive or disjunctive features. Each f_i approximates a conjunction when weights w_{ij} are very large. In the limit, f_i will be 1 iff the conjunct is true. f_i can also represent disjuncts using large negative weights, along with a negative weight for a "parent" feature f_k, w_{ki}. The negative weight w_{ki} turns the conjunction into a disjunction

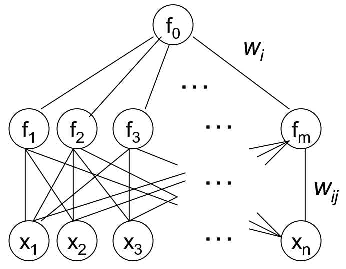

Figure 1. Simple propositional RRF network structure. This structure can represent any Markov random field with up to m conjunctive or disjunctive features, as well as many more complex distributions.

just as negation does in De Morgan's laws. However, one can also move beyond conjunction and disjunction to represent m-of-n concepts, or even more complex distributions where the feature values are given non-uniform weights.

Note that features with small weights have little effect. Therefore, instead of using heuristics or search to determine which attributes should appear in which feature, we can include *all* predicates and let weight learning sort out which attributes are relevant for which feature. This is similar to learning a neural network by initializing it with small, random values. Since the network can represent any logical formula, there is no need to commit to a specific structure ahead of time. An example of such an RRF structure is shown in Figure 3.1. This is an attractive alternative to the traditional inductive methods used for learning MRF features.

An RRF can be seen as a type of multi-layer neural network, in which the node function is exponential (rather than sigmoidal) and the network is trained to maximize joint likelihood. Unlike in neural networks, the random variables are all inputs, not inputs or outputs. The output is the likelihood of the random variables' joint configuration. In other ways, an RRF resembles a Boltzmann machine, but with the greater flexibility of multiple layers and learnable using a variant of the back-propagation algorithm. RRFs have no hidden variables to sum out, since all nodes in the network have deterministic values.

3.2. Relational RRFs

In the relational case, relations over an arbitrary number of objects take the place of a fixed number of variables. To allow parameter tying across different groundings, we use parameterized features, or *parfeatures*. We represent the parameter tuple as a vector, $\vec{g}$, whose size depends on the arity of the parfeature. (Note that $\vec{g}$ is a vector of logical variables—i.e., arguments to predicates—as opposed to the random Boolean variables $\mathbf{x}$—ground atoms—that represent a state of the world.) We use subscripts to distinguish among parfeatures with different parameterizations, e.g. $f_{i,\vec{g}}(\vec{x})$ and $f_{i,\vec{g'}}(\vec{x})$ represent different groundings of the ith parfeature.

Each RRF parfeature is defined in one of two ways:

$$f_{i,\vec{g}}(\mathbf{x}) \;=\; R_i(g_{i_1}, \ldots, g_{i_k}) \quad \text{(base case)}$$

$$f_{i,\vec{g}}(\mathbf{x}) \;=\; \frac{1}{Z_i} \exp\left(\sum_j w_{ij} \sum_{\vec{g'}} f_{j,\vec{g},\vec{g'}}(\mathbf{x}) \right) \quad \text{(recursive case)}$$

The base case is straightforward: it simply represents the value of a ground relation (as specified by $\mathbf{x}$). The grounding of the relation depends on the parameters of the parfeature. The recursive case sums the weighted values of all child parfeatures. Each parameter g_i of a child parfeature is either a parameter of the parent feature ($g_i \in \vec{g}$) or a parameter of a child feature that is summed out and does not appear in the parent feature ($g_i \in \vec{g'}$). (These $\vec{g'}$ parameters are analogous to the parameters that appear in the body but not the head of a Horn clause.) Just as sums of child features act as conjunctions, the summations over $\vec{g'}$ parameters act as universal quantifiers with Markov logic semantics. In fact, these generalized quantifiers can represent m-of-all concepts, just as the simple feature sums can represent m-of-n concepts.

The relational version of a recursive random field is therefore defined as follows:

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

where X is the set of all ground relations (e.g., $R(A, B)$, $S(A)$), $\mathbf{x}$ is an assignment of ground relations to truth values, f_0 is the root recursive parfeature (which, being the root, has no parameters). Since f_0 is a recursive parfeature, it is normalized by the constant Z_0 to ensure a valid probability distribution. (As in the propositional case, all other Z_i's can be absorbed into the weights of their parent features, and may therefore be normalized in any convenient way.)

Any relational RRF can be converted into a propositional RRF by grounding all parfeatures and expanding all summations. Each distinct grounding of a parfeature becomes a distinct feature, but with shared weights.

3.3. RRF Example

To clarify these ideas, let us take the example knowledge base from Richardson and Domingos (2006). The domain consists of three predicates: Smokes(g) (g is a smoker); Cancer(g) (g has cancer); and Friends(g, h) (g is a friend of h). We abbreviate these predicates as Sm(g), Ca(g), and Fr(g, h), respectively.

We wish to represent three beliefs: (i) smoking causes cancer; (ii) friends of friends are friends (transitivity of friendship); and (iii) everyone has at least one friend who smokes. (The most interesting belief from Richardson and Domingos (2006), that people smoke if and only if their friends do, is omitted here for simplicity.) We demonstrate how to represent these beliefs by first converting them to first-order logic, and then converting to an RRF.

One can represent the first belief, "smoking causes cancer," in first-order logic as a universally quantified implication: $\forall g : \text{Sm}(g) \Rightarrow \text{Ca}(g)$. This implication can be rewritten as a disjunction: $\neg\text{Sm}(g) \vee \text{Ca}(g)$. From De Morgan's laws, this is equivalent to: $\neg(\text{Sm}(g) \wedge \neg\text{Ca}(g))$, which can be represented as an RRF feature:

$$f_{1,g}(\mathbf{x}) = \frac{1}{Z_1}\exp(w_{1,1}\text{Sm}(g) + w_{1,2}\text{Ca}(g))$$

where $w_{1,1}$ is positive, $w_{1,2}$ is negative, and the feature weight $w_{0,1}$ is negative (not shown above). In general, since RRF features can model conjunction and disjunction, any CNF knowledge base can be represented as an RRF.

A similar approach works for the second belief, "friends of people are friends." In first-order logic: $\forall g : \text{Fr}(g,h) \wedge \text{Fr}(h,i) \Rightarrow \text{Fr}(g,i)$. This is easily rewritten as a disjunction and turned into the following RRF feature:

$$\begin{aligned} f_{2,g,h,i}(\mathbf{x}) &= \frac{1}{Z_2}\exp(w_{2,1}\text{Fr}(g,h) + \\ &\quad w_{2,2}\text{Fr}(h,i) + w_{2,3}\text{Fr}(g,i)) \end{aligned}$$

The first two beliefs are also handled well by Markov logic networks. The key advantage of recursive random fields is in representing more complex formulas. The third belief, "everyone has at least one friend who smokes," is naturally represented by nested quantifiers: $\forall g : \exists h : \text{Fr}(g,h) \wedge \text{Sm}(h)$. This is best represented as an RRF feature that references a secondary feature:

$$f_{3,g}(\mathbf{x}) = \frac{1}{Z_3}\exp\left(\sum_h w_{3,1} f_{4,g,h}(\mathbf{x})\right)$$

$$f_{4,g,h}(\mathbf{x}) = \frac{1}{Z_4}\exp(w_{4,1}\text{Fr}(g,h) + w_{4,2}\text{Sm}(h))$$

Note that in RRFs this feature can also represent a distribution over the number of smoking friends each person has, depending on the assigned weights. It's possible that, while almost everyone has at least one smoking friend, many people have at least two or three. With an RRF, we can actually learn this distribution from data.

This third belief is very problematic for an MLN. First of all, in an MLN it is purely logical: there's no change in probability with the number of smoking friends once that number exceeds one. Secondly, MLNs do not represent the belief efficiently. In an MLN, the existential quantifier is converted to a very large disjunction:

$$(\text{Fr}(g,A) \wedge \text{Sm}(A)) \vee (\text{Fr}(g,B) \wedge \text{Sm}(B)) \vee \cdots$$

If there are 1000 objects in the database, then this disjunction is over 1000 conjunctions. Further, the MLN will convert this DNF into CNF form, leading to 2^{1000} CNF clauses from each grounding of this rule.

These features define a full joint distribution as follows:

$$\begin{aligned} P(\mathbf{X}=\mathbf{x}) = \tfrac{1}{Z_0}\exp\ \Big(&\textstyle\sum_g w_{0,1} f_{1,g}(\mathbf{x}) + \\ &\textstyle\sum_{g,h,i} w_{0,2} f_{2,g,h,i}(\mathbf{x}) \\ &\textstyle\sum_g w_{0,3} f_{3,g}(\mathbf{x})) \end{aligned}$$

Figure 2 diagrams the first-order knowledge base containing all of these beliefs, along with the corresponding RRF.

4. Inference

Since RRFs generalize MLNs, which in turn generalize finite first-order logic and Markov random fields, exact inference is intractable. Instead, we use MCMC inference, in particular Gibbs sampling. This is straightforward: we simply sample each unknown ground predicate in turn, conditioned on all other ground predicates. The probability of a particular ground predicate may easily be computed by evaluating the relative likelihoods when the predicate is true and when it is false.

We speed this up significantly by caching feature sums. When a predicate is updated, it notifies its parents of the change so that only the necessary values are recomputed.

Our current implementation of MAP inference uses iterated conditional modes (ICM) (Besag, 1986), a simple method for finding a mode of a distribution. Starting from a random configuration, ICM sets each variable in turn to its most likely value, conditioned on all other variables. This procedure continues until no single-variable change will further improve the likelihood. ICM is easy to implement, fast to run, and guaranteed to converge. Unfortunately, it has no guarantee of converging to the most likely overall configuration. Possible improvements include random restarts, simulated annealing, or other ways of adding noise. MaxWalkSAT (Kautz et al., 1996) has been successfully applied to MAP inference in MLNs (Singla & Domingos, 2005), and in the future we may be able to adapt MaxWalkSat to RRFs as well.

We also use ICM to find an initial state for the Gibbs sampler. By starting at a mode, we significantly reduce the burn-in time and achieve better predictions sooner.

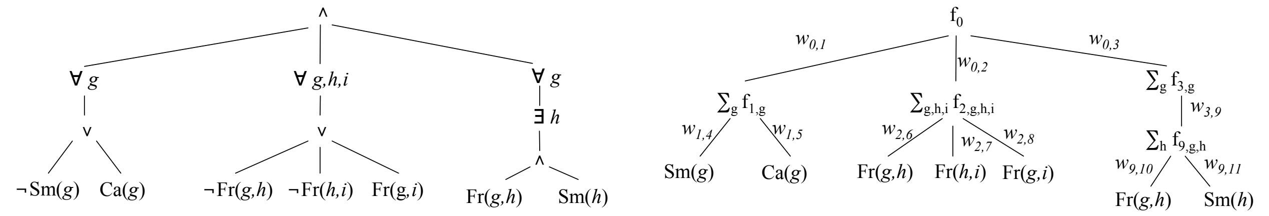

Figure 2. Comparison of first-order logic and RRF structures. The RRF structure closely mirrors that of first-order logic, but ANDs and ORs are replaced by weighted sums.

5. Learning

Given a particular RRF structure and initial set of weights, we learn weights using a novel variant of the back-propagation algorithm. As in traditional back-propagation, the goal is to efficiently compute the derivative of the loss function with respect to each weight in the model. In this case, the loss function is not the error in predicting the output variables, but rather the joint log likelihood of all variables. We must also consider the partition function for the root feature, Z_0. For these computations, we extract the $1/Z_0$ term from f_0, and use f_0 refer to the unnormalized feature value.

We begin by discussing the simpler, propositional case. We abbreviate $f_i(\mathbf{x})$ as f_i for these arguments. The derivative of the log likelihood with respect to a weight w_{ij} consists of two terms:

$$\frac{\partial \log P(\mathbf{x})}{\partial w_{ij}} = \frac{\partial \log(1/Z_0 f_0)}{\partial w_{ij}} = \frac{\partial \log(f_0)}{\partial w_{ij}} - \frac{\partial \log(Z_0)}{\partial w_{ij}}$$

The first term can be evaluated with the chain rule:

$$\frac{\partial \log(f_0)}{\partial w_{ij}} = \frac{\partial \log(f_0)}{\partial f_i} \frac{\partial f_i}{\partial w_{ij}}$$

From the definition of f_i (including the normalization Z_i):

$$\frac{\partial f_i}{\partial w_{ij}} = f_i \left(f_j - \frac{1}{Z_i} \frac{\partial Z_i}{\partial w_{ij}} \right)$$

From repeated applications of the chain rule, the $\partial \log(f_0)/\partial f_i$ term is the sum of all derivatives along all paths through the network from f_0 to f_i. Given a path in the feature graph $\{f_0, f_a, \ldots, f_k, f_i\}$, the derivative along that path takes the form $f_0 w_a f_a w_b f_b \cdots w_k f_k w_i$. We can efficiently compute the sum of all paths by caching the per-feature partials, $\partial f_0/\partial f_i$, analogous to back-propagation.

The second term, $\partial \log(Z_0)/\partial w_{ij}$, is the expected value of the first term, evaluated over all possible inputs $\mathbf{x}'$. Therefore, the complete partial derivative is:

$$\frac{\partial \log P(\mathbf{x})}{\partial w_{ij}} = \frac{\partial \log(f_0(\mathbf{x}))}{\partial w_{ij}} - E_{\mathbf{x}'} \left[\frac{\partial \log(f_0(\mathbf{x}'))}{\partial w_{ij}} \right]$$

where the individual components are evaluated as above.

Since explicitly computing the expectation is typically intractable, we approximate it using Gibbs sampling. For discriminative learning, an alternate approximation is the MAP state, as proposed by Collins (2002) and recently applied to MLNs (Singla & Domingos, 2005). The latter approach is likely to work better when more evidence is present.

In practice, we find that a small number of iterations of Gibbs sampling (< 100) per step of gradient descent is sufficient. In each iteration, we resample all of the random variables. While this will not sample all modes, it is usually enough to determine the rough direction of the gradient, a method Hinton (2002) refers to as "contrastive divergence." Since convergence can take hundreds or thousands of iterations, making each step as efficient as possible is quite important.

To learn a relational RRF, we use the domain to instantiate a propositional RRF with tied weights. The number of features as well as the number of children per feature will depend on the number of objects in the domain. Instead of a weight being attached to a single feature, it is now attached to a set of groundings of a parfeature. The partial derivative with respect to a weight is therefore the sum of the partial derivatives with respect to each instantiation of the shared weight.

6. RRFs vs. MLNs

Both RRFs and MLNs subsume probabilistic models and first-order logic in finite domains. Both can be trained generatively or discriminatively using gradient descent, either to optimize log likelihood or pseudo-likelihood. For both, when optimizing log likelihood, the normalization constant Z_0 is approximated using the most-likely explanation or Gibbs sampling.

Any MLN can be converted into a relational RRF by translating each clause into an equivalent parfeature. With sufficiently large weights, a parfeature approximates a hard conjunction or disjunction over its children. However,

	MLNs	RRFs
Non-relational MRF	Yes	Yes
Deterministic KB	Yes	Yes
Soft conjunction	Yes	Yes
Soft universal quant.	Yes	Yes
Soft disjunction	No	Yes
Soft existential quant.	No	Yes
Soft nested formulas	No	Yes

Table 1. Concepts that each type of model can efficiently represent.

when its weights are sufficiently distinct, a parfeature can take on a different value for each configuration of its children. This allows RRFs to compactly represent distributions that would require an exponential number of clauses in an MLN. Any RRF can be converted to an MLN by flattening the model, but this will typically require an exponential number of clauses. Such an MLN would be intractable for learning or inference.

A brief comparison of the representational power of MLNs and relational RRFs is provided in Table 1.

In addition to being "softer" than an MLN clause, an RRF parfeature can represent many different MLN clauses simply by adjusting its weights. This makes RRF weight learning more powerful than MLN structure learning: an RRF with $n + 1$ recursive parfeatures (one for the root) can represent any MLN structure with up to n clauses, as well as many distributions that an n-clause MLN cannot represent.

This leads to new alternatives for structure learning and theory revision. In a domain where little background knowledge is available, an RRF could be initialized with small random weights and still converge to a good statistical model. This is potentially much better than MLN structure learning, which constructs clauses one predicate at a time, and must adjust weights to evaluate every candidate clause.

When background knowledge is available, we can revise the existing theories as well as add new ones. We begin by initializing the RRF to the background theory, just as in MLNs. However, in addition to the known dependencies, we can add dependencies on other parfeatures or predicates with very small weights. If other dependencies are relevant, they can gain larger weights through learning. If any dependencies are irrelevant, their weights can be reduced to negligible values through learning. In this way, RRFs can do theory revision analagous to what the KBANN system does using neural networks (Towell & Shavlik, 1994). In contrast, MLNs can only do theory revision through discrete search.

7. Experiments

We now present a preliminary empirical demonstration of how RRFs can be learned.

The datasets we used consisted of five unary attributes and one binary relation, grounded over 100 objects. Each object was randomly assigned to one of two clusters. The probability of each attribute depends only on the object's cluster. The relation is true with probability 0.5 for two objects within the same cluster and 0.05 for two objects in different clusters. The relation is always symmetric: $R(A, B) \iff R(B, A)$.

We generated 5 datasets in this manner, each time with different attribute probabilities. We fixed one attribute, A_0, to be always true in the first cluster and always false in the second cluster, and used this as our test predicate.

A clustering model can be naturally represented as a noisy DNF, where each clause in the disjunction represents a cluster. The clauses, being noisy, do not exactly specify the attribute values, but rather a distribution over them. We initialized our RRFs with the following model:

$$\begin{aligned} f_0(\mathbf{x}) &= \exp\left(-1\sum_{g,h} f_{1,g,h}(\mathbf{x})\right) \\ f_{1,g,h}(\mathbf{x}) &= \exp(-f_{2,g,h}(\mathbf{x}) - f_{3,g,h}(\mathbf{x}) + R(g,h)) \\ f_{2,g,h}(\mathbf{x}) &= \exp(0.01A_0(g) + 0.01A_0(h) + \ldots) \\ f_{3,g,h}(\mathbf{x}) &= \exp(0.0A_0(g) + 0.0A_0(h) + \ldots) \end{aligned}$$

Each equality defines a parfeature as a function of other parfeatures or predicates. The top-level negation creates a disjunction over three cases: g and h are are not related via $R(g, h)$; g and h are in the first cluster; or g and h are in the second cluster. Within the clusters, the weights give a slight bias towards objects having similar values.

For the per-parfeature normalization constants Z_i, we tried normalizing over all values, so that each feature remains a valid probability distribution. However, this causes the feature to take values in an overly narrow range, leading to potential numeric problems. Instead, we normalized based on the maximum possible value of the feature, so that the effective range of each f_i remains approximately $[0, 1]$. Since the derivative of the max function is discontinuous, we used a continuous variant:

$$Z_i = \exp\left(\sum_j w_{ij}\sigma(w_{ij})\right)$$

where $\sigma(w_{ij}) = 1/(1+\exp(-w_{ij}))$, the sigmoid function.

We trained RRF models for 1000 iterations of gradient descent, each step taking 1-3 seconds. We used a Gaussian

Trial	NB	RRF (train)	RRF (PLL)	RRF (LL)
1	-0.2043	-0.0625	-0.0133	-0.0131
2	-0.3578	-0.0001	-0.0017	-0.0020
3	-0.2431	-0.0006	-0.2628	-0.3648
4	-0.2136	-0.0002	-0.5575	-0.5748
5	-0.4914	-0.0001	-0.0018	-5.6088

Table 2. Results for a simple two-cluster relational model. NB is the true model, ignoring relational information. The next three columns are the performance of a trained RRF on the training data, on the test data, and on the test data with all query predicates hidden. The last case is what the learning procedure attempts to optimize.

prior with mean zero and $\sigma^2 = 100$ on each weight. In Table 7, we present the results on each dataset. The first column lists the per-predicate log likelihood of A_0 using a non-relational naive Bayes model. The parameters for the model are the same as those used to generate the data, except that it ignores the relation. This represents the best possible model that disregards all relational information. The second column shows per-predicate RRF pseudo-log-likelihood (i.e., conditional log-likelihood of each predicate given its Markov blanket in the data). This is the easiest case for RRFs, since the training and test data are the same, and all but one grounding of A_0 were known at inference time. The next column shows pseudo-log-likelihood performance on a test set, independently generated from the same distribution as the training set. RRFs convincingly outperform naive Bayes in trials 1 and 2. Trials 3 and 4 show evidence of overfitting.

The final column shows the average log likelihood of the test predicates. 100 variables were unknown at inference time. We approximated the likelihood using 1000 samples of each variable from each of 10 chains of Gibbs sampling. In total, each query atom was sampled 10,000 times. One problem with Gibbs sampling over cluster models is the tendency to become trapped in local minima. This seems to be the case with Trial 5. A more sophisticated MCMC technique (like simulated tempering or slice sampling) may overcome this.

We also experimented with MLNs on this dataset. However, we were unable to obtain good results: the models we learned consistently fell into poor modes and mispredicted most groundings. We are currently investigating this.

Our experiments cannot yet prove that RRFs are practical, but they do show potential and highlight areas of future work.

8. Conclusion

Recursive random fields overcome some salient limitations of Markov logic. While MLNs only model uncertainty over conjunctions and universal quantifiers, RRFs also model uncertainty over disjunctions and existentials, and thus achieve a deeper integration of logic and probability. Inference in RRFs can be carried out using Gibbs sampling and iterated conditional modes, and weights can be learned using a variant of the back-propagation algorithm.

The main disadvantage of RRFs relative to MLNs is reduced understandability. One possibility is to extract MLNs from RRFs with techniques similar to those used to extract propositional theories from KBANN models. Another important problem for future work is scalability. Here we plan to adapt many of the MLN optimizations to RRFs. These include using subsampling to compute sufficient statistics (Kok & Domingos, 2005), adapting MaxWalkSAT (Singla & Domingos, 2005), and adapting slice sampling (Poon & Domingos, 2006).

Most importantly, we intend to apply RRFs to real datasets to better understand how they work in practice, and to determine if their greater representational power results in better models.

Acknowledgments

The authors thank Stanley Kok for help with the experiments. This work was supported by a National Science Foundation Graduate Research Fellowship awarded to the first author, DARPA grant FA8750-05-2-0283 (managed by AFRL), DARPA contract NBCH-D030010, NSF grant IIS-0534881, and ONR grant N00014-05-1-0313. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, NSF, ONR, or the United States Government.

References

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B*, *48*, 259–302.

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

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

Kautz, H., Selman, B., & Jiang, Y. (1996). A general stochastic approach to solving problems with hard and soft constraints. In D. Du, J. Gu and P. M. Pardalos (Eds.), *The satisfiability problem: Theory and applica-*

tions, 573–586. New York, NY: American Mathematical Society.

Kok, S., & Domingos, P. (2005). Learning the structure of markov logic networks. *Proceedings of the Twenty-Second International Conference on Machine Learning* (pp. 441–448). Bonn, Germany: ACM Press.

Poon, H., & Domingos, P. (2006). Sound and efficient inference with probabilistic and deterministic dependencies. *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Boston,MA: AAAI Press.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*, 107–136.

Singla, P., & Domingos, P. (2005). Discriminative training of Markov logic networks. *Proceedings of the Twentieth National Conference on Artificial Intelligence* (pp. 868–873). Pittsburgh, PA: AAAI Press.

Towell, G. G., & Shavlik, J. W. (1994). Knowledge-based artificial neural networks. *Artificial Intelligence*, *70*, 119–165.