Machine Learning

Fast Learning of Relational Dependency Networks

--Manuscript Draft--

Manuscript Number:	MACH-D-14-00294R1
Full Title:	Fast Learning of Relational Dependency Networks
Article Type:	S.I. : ILP 2014
Keywords:	Relational Dependency Network; Bayesian network; log-linear model; functional gradient boosting; multi-relational data
Corresponding Author:	Zhensong Qian Simon Fraser University CANADA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Simon Fraser University
Corresponding Author's Secondary Institution:	
First Author:	Oliver Schulte, PhD
First Author Secondary Information:	
Order of Authors:	Oliver Schulte, PhD
	Zhensong Qian
	Arthur E. Kirkpatrick, PhD
	Xiaoqian Yin
	Yan Sun
Order of Authors Secondary Information:	
Funding Information:	
Abstract:	A Relational Dependency Network (RDN) is a directed graphical model widely used for multi-relational data. These networks allow cyclic dependencies, necessary to represent relational autocorrelations.We describe an approach for learning both the RDN's structure and its parameters, given an input relational database: First learn a Bayesian network (BN), then transform the Bayesian network to an RDN. Thus fast Bayesian network learning translates into fast RDN learning. The BN-to-RDN transform comprises a simple, local adjustment of the Bayesian network structure and a closed-form transform of the Bayesian network parameters. This method can learn an RDN for a dataset with a million tuples in minutes. We empirically compare our approach to a state-of-the-art RDN learning approach that applies functional gradient boosting, using six benchmark datasets. Learning RDNs via BNs scales much better to large datasets than learning RDNs with current boosting methods, and provides competitive accuracy in predictions.
Response to Reviewers:	Reply to Reviews We would like to thank cordially the reviewers and the guest editors for their time and consideration. Putting together a special issue with complex papers takes a lot of time; it's a great service for the community. The reviewer suggestions were very valuable to us. We believe we have addressed them all, and that this contributed to both the clarity and the substance of the paper. As directed by the editors, we have prepared a point-by-point reply. To make it easy to process our reply and track our changes, we have structured the reply as follows. 1) Point-by-point replies inserted into the original reviews. 3) Detailed discussion about reproducing the results of the boosting experiments from previous papers, answering questions raised by reviewer #1.

From: "Guest Editors of ILP 2014" <em@editorialmanager.com>
To: "Zhensong Qian" <zqian@sfu.ca>
Sent: Tuesday, February 17, 2015 4:50:17 AM
Subject: Decision on your manuscript #MACH-D-14-00294

CC: d.margin@comcast.net, jesse.davis@cs.kuleuven.be, jan.ramon@cs.kuleuven.be

Dear Mr. Zhensong Qian,

The reports from the reviewers of your manuscript, "Fast Learning of Relational Dependency Networks", which you submitted to Machine Learning, have now been received.

Based on the advice received, your manuscript requires major revisions in order to be acceptable for publication. It is the policy of Machine Learning not to accept manuscripts requiring major revisions. Instead, authors choosing to revise their manuscript according to the reviewers' comments must resubmit a revised manuscript, which usually will be reassessed by the same action editor.

When preparing your revised manuscript, you should carefully consider the reviewers' comments, and submit a point wise list of responses to the comments. Your list of responses should be uploaded as a separate file in addition to your revised manuscript.

Among other points, we would encourage you to address the following:

1. Improving some of the motivation for tackling the problem in this manner.

-We added a “Motivation” paragraph to the introduction. This discusses why using Bayesian network learning to construct relational dependency networks allows us to combine the strengths of each model type for relational data. These strengths have been discussed in previous papers, and we provide references. We also discuss a predecessor paper that converts Bayesian networks to Markov networks with a similar motivation.
- Our specific development of this general idea introduces a novel log-linear equation for defining dependency network parameters. In Section 4.2 we discuss in detail three key properties that in our view motivate the log-linear equation.

2. Discussing the implications of the learning bias.

The language bias (first-order variables only, no constants, no aggregate functions) does not apply to our main contribution, the DN-to-BN conversion. It applies only to the previously existing structure learning method we use. We agree that it is important to clarify this, so we mention this point now explicitly when we introduce Parametrized Bayesian network in Section 2.2. We also discuss it again when we describe the structure learning method in Section This language bias is very common in structure learning, and we give references to previous work with the same bias. We also discuss it again in the structure learning Section 6.2, and contrast it with the gradient boosting methods which do not impose the language bias.

3. Improving the empirical evaluation section along the lines suggested by the reviewers.

Please see detailed replies below.

4. Cleaning up some of the technical details.

Please see detailed replies below.

Finally, as this is a special issue we would appreciate receiving your revisions within 12 weeks. Please email us and confirm if this time frame seems feasible or not.

Please make sure to submit your editable source files (i. e. Word, TeX).

In order to submit your revised manuscript electronically, please access the following web site:

http://mach.edmgr.com/

Your username is: zqian@sfu.ca

To access your password, please complete the following steps:

1. Open the URL: http://mach.edmgr.com/
2. Click the LOGIN button on the banner.
3. Click "Send Username/Password".
4. Complete the required information (First Name, Last Name, Email Address).
5. Click "Send Username and Password".

Your Password will be sent to you by email.

Please click "Author Login" to submit your revision.

Thank you.

Best regards,

Jesse Davis and Jan Ramon
Guest Editors of ILP 2014
Guest Editor
Machine Learning

COMMENTS FOR THE AUTHOR:

Reviewer #1: This paper considers the problem of learning Relational Dependency Networks. To this effect, the authors propose a three step process of first learning a Bayes net, then converting it into a dependency and finally, lifting this to a RDN. The claim is that this can scale to very large amounts of data. The experimental results compared to a recent boosting algorithm backs up this claim.

The idea in the paper is quite simple and the paper is written in a clear manner to explain "what" happens in the algorithm. So from that perspective the paper is pretty well written. The experimental results are quite compelling (though I have some questions) to back up the key claims in the paper. Things I like about the paper are the simplicity of the idea, its relatively clear explanation, nice related work section that discusses the different methods quite diligently and finally compelling experimental results.

Thank you for the positive feedback. Much appreciated.

I do have some key issues with the paper that need to clarified/fixed before fully accepting the paper.

My key issue with the paper is the motivation itself. I am quite baffled with the claim that learning Bayes net is somehow easier than learning a dependency network. More pertinently learning Bayes net can scale somehow more than a DN.

This is a special feature of relational learning as compared to propositional learning from i.i.d. data. In relational data, data access is relatively more expensive than with i.i.d. data, because evaluating complex conjunctive patterns requires combining information from different relations (e.g., table joins in database terminology). In typical dependency network learning methods, model evaluation requires iterating over all data points and testing the predictions of the model for each data point. In contrast, Bayesian network models can be evaluated in closed-form by using sufficient statistics only (aggregate counts). This is because for both i.i.d. and for relational data, there is a

	theorem that provides a closed-form for maximum likelihood parameter estimates. Basically this means that the cost of Bayesian network learning is dominated by the cost of counting the number of groundings that satisfy a formula in the input database. This is not an easy problem, but it can generally be solved much faster than iterating over ground facts in the database, see discussion and references in Section 4.3. We have added explicit discussion of scalability in the introduction in the motivation subsection. We have added a separate section on Bayesian network learning (Sec. 6) and discuss the scalability of Bayesian network learning there. We have put this in a separate section because Bayesian network learning is not the main contribution of our paper, which is the Bayes net-to-dependency net conversion. We hope that this strikes a good balance between reviewing relevant and interesting information from previous work vs. allowing readers to focus on our main contribution. We all know that learning a full BN is NP-Complete (Chickering 96). In such cases, claiming that learning a BN with acyclicity constraints is somehow easier than learning several local models (boosted or not) is quite unintuitive. While experimental results show some idea, it is clear that the issue is with the number of predicates. If the number of variables increases, clearly the number of parameters increases as well and clearly this method cannot scale. This is now discussed in the Bayesian network learning section, and also in 8.4. where we discuss BN learning and gradient boosting. The BN model search is indeed more complex than learning several local models. But the main cost for relational learning is data access. BN learning can be scaled up to large numbers of predicates (nodes) using methods like Friedman's Sparse Candidate algorithm. In our experiments, RDN-Bayes is still fast on databases with many predicates like IMDb (17) and Hepatitis (19). The second (and probably more confusing issue) for me is that why learn a RDN when you have a chance of learning a RBN. RBNs are fairly better interpretable and much tighter in semantics. Not much work has been done in SRL for learning directed models effectively. If your method can learn a BN successfully and in larger scale why bother with converting it to an approximate model instead of lifting it directly? This motivation is unclear and confusing to me. So this process and the lack of motivation is quite confusing to me. Thank you for raising this issue. This is another way in which relational and propositional data differ. In relational data, cyclic dependencies are important, and these are notoriously difficult to model with Bayesian networks. We mention this in the introduction under "Motivation" and give references to previous papers that discuss the difficulties of using Bayesian networks when the domain contains cyclic dependencies. So the inference advantage is not that the approximate model is better, but that it is difficult to do or even define inference with Bayesian networks when there are cyclic dependencies. In addition to providing additional motivation for the general idea of using a hybrid approach, we also discussed at more length the motivation for our specific log-linear equation (Sec. 4.2). What is the learning algorithm for BNs? This needs more explanation. What are the properties of the data that will make this learning effective? Why is a NP-complete problem an easy one for you? Is this assuming independence of variables or is it the fact that you consider a small number of predicates? This needs to be better explained for things to be clear. There is now a separate section devoted to Bayesian network learning (Section 6). This reviews the previously existing BN learning algorithm we use. It also reviews previous results on the scalability of Bayesian network learning, including the NP-hardness. Which brings me to the experimental setting. If the data sets used a smaller number of predicates to learn from, I can easily see why boosting will not perform as well. But the numbers reported in this work are well-below the ones reported in the prior work that the authors have cited. Why not include another part that shows the scale with the

number of predicates?

We wonder if there is a misunderstanding here. As Table 2 shows, the datasets in our study contain as many as 17,18,19 predicates. This is a larger number than in most statistical-relational work, including the boosting papers. The boosting paper seems to use at most 15 predicates, including equality predicates. The easiest way to check this is to download the datasets from the Boostr website http://pages.cs.wisc.edu/~tushar/Boostr/contact.html.

In your learning of the parameters, it is unclear how the probabilities are propagated to newly introduced bi-directional links. For existing links, I can see a weighted count being the new potential function. It appears that the Gibbs sampling comes in when you are computing the fractions (feature functions), but this is not explained quite clearly. This explanation must be improved. Back to the original question, if the original link is B->A in the BN, when you create a A->B link, how can you simply use the parameters of the original link? This needs more explanation and is unclear. Simply stating that this is same as moralizing does not seem sufficient in the case of bi-directional case. So please improve the explanation of this part.

We added a discussion of this point in Section 4.2. where we discuss properties of the log-linear equation (see “generalizing the propositional case”). This is the same as in the propositional case, in the sense that the propositional network local distribution equation also uses the B->A probability when predicting A given B.

Also please do explain if the equation on Page 8, lines 12/13 are the ones obtained by Gibbs sampling.

Referring to family counts and proportions, in our experiments we use exact counting methods to obtain these, not Gibbs sampling. The log-linear equation can be applied no matter how the counts are obtained. We added a remark to this effect after the equation for the family proportion.

Another issue seems to be inventing new names. Why call something as Gibbs conditional probabilities? What does this even mean? These are local distributions over a particular variable and you perform Gibbs sampling in this space because these local distributions may not be consistent. This terminology of Gibbs conditional probabilities only adds to more confusion and does not clarify anything in particular.

Good point, we changed to “local distribution” throughout.

The experimental setup needs to be better explained. The results are explained clearly. I am not sure the claim that previous methods work for small data sets is quite correct. if you mean small in terms of the number of instances, then a better citation to the original papers may be needed. I checked the original paper and they handle the 0.1M data set size that you mention.

Referring to MovieLens, the Natarajan et al. 2012 MLJ paper states that

“Our next data set is the Movie Lens data set [50]. The dataset was created by randomly selecting a subset of 100 users and 603 movies.”

So their experiments were done on a smaller subset than ours (941 users, 1682 Movies). For more details please see the reproducibility section.

So how is this claim justified? Their AUC-PR values seem much higher in the original paper. Can you speculate why this difference?

We agree that reproducing results is important, so we tried very hard to reproduce the results in the original boosting papers, and we checked this again given your comments. There are many details here about which parts of which datasets were used, which system settings etc. We describe these details at the end of our reply because they are too involved for the point-by-point format.

While the claim of interpretability of a set of regression trees is a valid criticism and in fact a key reason why boosting must not be the first choice for several problems, I am not sure I agree with the idea about extending learning to more than two values. We all know that many of these n-class learning problems can be seen as n-binary learning problems and so this does not seem a big issue.

We have reduced this point to one sentence in a less prominent place (item (3) in Section 7.4). We agree that there are methods for reducing n-class learning to two-class learning, but a boosting user would still have to decide which of these to use. We could remove this point altogether, our main argument really does not hinge on this.

Over all, I think the paper has promise in terms of the idea. It lacks motivation, details on when things work and certainly better experimental justification.

Thank you again for your time and the constructive criticism. As a final high-level comment, we'd like to clarify that our argument is not that our Bayes net conversion approach always gives better models than gradient boosting. Gradient boosting is powerful and elegant. But just like there are multiple methods for structure learning for propositional data, or for Markov Logic networks in relational data, there is room for multiple methods for relational dependency networks. Our method is completely novel, and we believe that both theoretical considerations and empirical results show that it has new strengths (and limitations) compared to gradient boosting. Our view is that ultimately the best system will combine boosting ideas with fast structure learning, and we make concrete suggestions for achieving this (Section 8.4).

Reviewer #2: The paper shows how to learn relational dependency networks
via Bayesian networks. The experimental comparison compares
to state of the art and indicates competitive performance.

Thank you for your time and suggestions.

Overall, I like the paper. However, in its current form,
there are some downsides:

- no intuition provided
- unclear learning bias
- a little bit sloppy on some technical details

The paper is a little bit too loose about some technical details.
For instance, on page 2, you argue that dependency networks can
derive the joint distribution using Gibbs sampling. This, however,
is only true if you run an unordered Gibbs sampler, see

Yoshua Bengio, Eric Laufer, Guillaume Alain, Jason Yosinski:
Deep Generative Stochastic Networks Trainable by Backprop.
ICML 2014: 226-234

The two references you give [9,17] also touch upon this in that
they speak up pseudo Gibbs sampler. This sheets also some light
on Theorem 1. First if all, we may not need the proposed refinement
when sticking to an unordered Gibbs sample (see Bengio et al.).
Nevertheless, it is nice that since we start with a BN, we can
construct a consistent RDN.

This is a great reference and we were happy to add it (Section 5 after Theorem 1). We believe that it supports our approach. Theorem 1 gives a necessary and sufficient condition for the derived dependency network to be consistent. We note after Theorem 1 that this condition is restrictive and will not be met by most parametrized Bayesian networks learned from relational data. (This is a relational phenomenon, BN-2-DN conversion for propositional data produces a consistent dependency network as observed by Heckerman et al.). Stochastic Generative Networks provide a powerful new way to define a joint (stationary) distribution from inconsistent conditional

distributions, so this is a new way to do joint reasoning from the kind of relational dependency networks we learn in this paper.

While we are happy to add the reference, we have downplayed the discussion of inference in dependency networks in general because that is not the focus of our paper. Instead we focused on the motivation for our approach as suggested by the reviewers.

On page 2, you speak of functors. Technically speaking a functor is not a predicate as claimed.
Functors can only be used to form compound terms and not atoms.
Moreover, as far as I understand your notation, you are actually
not allowing for compound terms. Hence, I would suggest to just call
this a predicate and subsequently of atoms. That is atoms
denote parameterized RVs. On page 3, you say that one just has
to ground the parents and the child. While this is technically true,
the success of relational dependency networks critically depends on
using aggregation functions (min, max, count, or any other „function"
defined in the background knowledge). With this, however, one cannot
say anymore that we simply have to ground parents and children.

Thank you for this observation. We have not introduced any new notation or terminology but instead followed Poole's 2003 paper and the survey by Kimmig et al. 2014.

We discuss aggregate functions along with the use of constants as part of the language bias as suggested by referee 1, in Section 6.2 and in Section 2.2. We also give references to previous work with aggregate functions, such as that by Friedman, Getoor, deRaedt and Kersting. The main contribution of our paper, which is the BN-to-DN conversion, is a general procedure that can be applied with different languages for defining parametrized random variables. It is the previously existing Bayesian network structure learning algorithm (the LAJ algorithm) that is limited in the expressiveness of the patterns it searches. There is a new section 6 devoted entirely to discussing Bayes net learning, including the language bias of the LAJ implementation. In our experimental results, the log-linear model provided good predictive accuracy even without aggregate functions. Using Bayesian network learners that learn with aggregate functions (cf. Section~\ref{sec:rdns}) should improve predictive accuracy even further, we mention this now in Section 6. We are not aware of any available implementation/code that offers BN network structure learning with relational aggregates.

More interestingly is the idea of learning a relational DN from
a (relational) BN. For the propositional case, there seems to be not
major benefits of doing so. Actually, the opposite is more appropriate,
see also

G. Hulten, D.M. Chickering, D. Heckerman. Learning Bayesian Networks from Dependency Networks: A Preliminary Study. In Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, Key West, FL, January 2003,

which should be cited. Hence, the authors should discuss this situation
since it is covered by RDN_Bayes.

We've added this reference. A similar point was raised by reviewer 1. It is perhaps easiest if we repeat our comments to reviewer 1. This is a special feature of relational learning as compared to propositional learning from i.i.d. data. In relational data, data access is relatively more expensive than with i.i.d. data, because evaluating complex conjunctive patterns requires combining information from different relations (e.g., table joins in database terminology). In typical dependency network learning methods, model evaluation requires iterating over all data points and testing the predictions of the model for each data point. In contrast, Bayesian network models can be evaluated in closed-form by using sufficient statistics only (aggregate counts). This is because for both i.i.d. and for relational data, there is a theorem that provides a closed-form for

	maximum likelihood parameter estimates. Basically this means that the cost of Bayesian network learning is dominated by the cost of counting the number of groundings that satisfy a formula in the input database. This is not an easy problem, but it can generally be solved much faster than iterating over ground facts in the database see discussion and references in Section 4.3. . We have added explicit discussion of scalability in the introduction in the motivation subsection. We have added a separate section on Bayesian network learning (Sec. 6) and discuss the scalability of Bayesian network learning there. We have put this in a separate section because Bayesian network learning is not the main contribution of our paper, which is the Bayes net-to-dependency net conversion. We hope that this strikes a good balance between reviewing relevant and interesting information from previous work vs. allowing readers to focus on our main contribution. Generally, the authors do not discuss the intuition underlying their approach. We have provided additional explicit motivation for the general idea of using a hybrid approach (see introduction). This, however, is critical to understand the conversion to RDNs starting on page 5 and consequently the experimental results. To be more precise, why do we consider the expected log-conditional probability? We have discussed at more length the motivation for our specific log-linear equation (Sec. 4.2). A relational dependency network can in principle have different parameters per ground atom. The proposed method, however, implicitly couples all ground atoms of a predicate (parameterized RV). Please note, I am not saying that this is a bad idea, but it is not clear to me whether this learning bias is always what we want; a discussion is in place. We mention explicitly that we use parameter tying, like other log-linear SRL models , with references. This is related to the another major downside: the learning of the Bayesian structure is not explained at all, instead it is referred to [23]. The authors should extend the current paper by a brief discussion of the learn-and-join algorithm, in particular w.r.t. to the „issue" raised above. As far as I read [23], only most general atoms are considered, i.e., all arguments are variables. Why is this important? Well the baseline approaches consider the considerably large search space where arbitrary atoms (arguments can be constants) are considered. In turn, it is not clear whether the novel learning approach gives the performance gain or the implicitly encoded learning bias (no constants in atoms). This makes learning considerably faster and provides a regularization, which may lead to better performance. At least this has to be discussed in detail. There is now a separate section devoted to Bayesian network learning (Section 6). This reviews the previously existing BN learning algorithm we use. It also reviews previous results on the scalability of Bayesian network learning, including the NP-hardness. It discusses the issues of language bias. These pertain to the previous BN learning methods, not to our new BN-to-DN conversion method. The movieLens experiments seem to support that it is mainly the

implicitly regularization since with large dataset size the boosting methods start to outperform the proposed RDN_Bayes in terms of CLL.
Please specify exactly the predicates you have used for the AUC comparison. Looks like Table 4 is presenting AUC averaged over different predicates.

From the paper “we evaluate accuracy on binary predicates (e.g., gender,Borders)”. We’ve changed this to “we evaluate accuracy on all binary predicates (e.g., gender,Borders)” to be more precise. Or did you mean “explicitly list all predicates for each dataset?”. We can do that if that improves the presentation, sure.

Also, do the running times include the time spend on learning the Bayes structure? Although I guess they are neglect able.

Table 2 shows: The total learning time for constructing a relational dependency network from an input database. This includes learning the structure and parameters of the Bayesian network.

The experimental results should also include the performance of the Bayes structure to see what the conversion to an RDN is giving in terms of performance gain.

This is a difference between the propositional case and the relational case. In the relational case, a Bayesian network structure does not by itself define an inference model that can be evaluated. It needs to be augmented with some kind of aggregation, e.g. using aggregates like count, max, exists etc, or with combining rules. See the reference below for discussion of this “multiple instantiation problem”. Depending on the kind of aggregation/combining rules, one needs to use different parameter learning algorithms. Comparing with more statistical-relational models is of course always good. But there isn’t really a straightforward way for just evaluating the BN structure produced by the learn-and-join algorithm. Plus there is the problem of defining BN inference in the presence of cyclic dependencies that we discuss in the introduction.

@ARTICLE{Natarajan2008,
  author = {Sriraam Natarajan and Prasad Tadepalli and Thomas G. Dietterich and Alan Fern},
  title = {Learning first-order probabilistic models with combining rules},
  journal = {Annals of Mathematics and Artifical Intelligence},
  year = {2008},
  volume = {54},
  pages = {223-256},
  number = {1-3},
  bibsource = {DBLP, http://dblp.uni-trier.de},
  ee = {http://dx.doi.org/10.1007/s10472-009-9138-5},
  file = {Natarajan2008.pdf:Natarajan2008.pdf:PDF},
  keywords = {statistical-relational}

Finally, please note that boosting is not creating discriminative models. They are just estimating RDNs, a joint model using the pseudo-loglikelhood factorization. And, you do not have to analyze an ensemble of regression trees. They can always collapsed after training is finished into a single tree if you like.

We removed references to discriminative models wrt boosting.

Nevertheless, the contributions are indeed interesting. The most important question is about the learning bias. That is, the authors should provide some intuition about what makes the learning better and faster.
The additional number of predicates used seems not to provide a complete answer, at least the Bayes structure is not factored in at all in this answer.

The Bayes structure shows which relationship chains carry probabilistic information and hence which predicates from tables other than the target table are relevant to the

target predicate. The ability to find complex patterns with longer relationship chains comes from the lattice search strategy, which in turn depends on the scalability of model evaluation to explore a complex space of relationship chains. We hope that adding the section on BN learning makes this clearer.

Thank you again for your time and the helpful observations. As a final high-level comment, we'd like to clarify that our argument is not that our Bayes net conversion approach always gives better models than gradient boosting. Gradient boosting is powerful and elegant. But just like there are multiple methods for structure learning for propositional data, or for Markov Logic networks in relational data, there is room for multiple methods for relational dependency networks. Our method is completely novel, and we believe that both theoretical considerations and empirical results show that it has new strengths (and limitations) compared to gradient boosting. Our view is that ultimately the best system will combine boosting ideas with fast structure learning, and we make concrete suggestions for achieving this (Section 8.4).

Reviewer #3: This paper presents a method for quickly learning relational dependency networks (RDNs) via first learning Bayesian networks (BNs).

The rationale for "compiling" to RDNs is to make use of their "advantages of inference" (page 1, line 30). However, these advantages are not apparent in the context of the empirical evaluation, which predicts the probability of a target ground predicate given all other ground predicates as evidence (page 11, line 16). In this setting, wouldn't the inference in BNs be easy too?
This is a way in which relational and propositional data differ. In relational data, cyclic dependencies are important, and these are notoriously difficult to model with Bayesian networks. So by "advantage for inference" we mean the ability to reason with cyclic dependencies in dependency networks. This was the motivation given by Neville and Jensen in their original relational dependency network paper. We discuss this in detail in a new subsection in the introduction under "Motivation" and give references to previous papers that discuss the difficulties of using Bayesian networks when the domain contains cyclic dependencies.

In addition to providing additional motivation for the general idea of using a hybrid approach, we also discussed at more length the motivation for our specific log-linear equation (Sec. 4.2).

To demonstrate the value of the paper's proposed algorithm, there should be experiments comparing the performance of the learned BNs to that of the RDNs (that are derived from the BNs). Only if the RDNs perform better, will the overhead of "compiling" them be justified.

Reviewer 2 raised a similar issue.
This is a difference between the propositional case and the relational case. In the relational case, a Bayesian network structure does not by itself define an inference model that can be evaluated. It needs to be augmented with some kind of aggregation, e.g. using aggregates like count, max, exists etc, or with combining rules. See the reference below for discussion of this "multiple instantiation problem". Depending on the kind of aggregation/combining rules, one needs to use different parameter learning algorithms. Comparing with more statistical-relational models is of course always good. But there isn't really a straightforward way for just evaluating the BN structure produced by the learn-and-join algorithm. Plus there is the problem of defining BN inference in the presence of cyclic dependencies that we mentioned.

@ARTICLE{Natarajan2008,
  author = {Sriraam Natarajan and Prasad Tadepalli and Thomas G. Dietterich and Alan Fern},
  title = {Learning first-order probabilistic models with combining rules},

journal = {Annals of Mathematics and Artifical Intelligence},
year = {2008},
volume = {54},
pages = {223-256},
number = {1-3},
bibsource = {DBLP, http://dblp.uni-trier.de},
ee = {http://dx.doi.org/10.1007/s10472-009-9138-5},
file = {Natarajan2008.pdf:Natarajan2008.pdf:PDF},
keywords = {statistical-relational}

Questions

* Page 5, Algorithm 1, line 8
By restricting F to containing only true relationships, is it true that that you are confining features to vary over values of attributes (e.g., gender(A)=M, gender(B)=M)? Why do you need to make such a restriction? And what happens if you don't?

Yes, predictive features vary over attributes only. The main problem that arises without this restriction is that links are typically very sparse. So there are many more negated links and they will overwhelm the information from actual links. In our friendship example, if a social network contains 10,000 users, and Sam has 100 friends, a feature like gender(sam)=M, NOT Friend(sam,B) will be instantiated 10,000-100 = 99,900 times. But this is a good question, and we have actually developed methods for trying to use negated links, but explaining them would complicate the paper quite a bit, certainly go beyond the ILP submission to which this was meant to be an extension. Since other statistical-relational methods use the same approach of using existing links only (references in the paper), we leave this issue for future work.

* Page 6, Definition 1.
The second summation ranges over all values of u_pa. However, when u is a child of T, u_pa contains T, and its value must be tied to T*=t. Could the authors make this condition clearer? (Table 1 illustrates this situation clearly.)

We added a note stating this constraint explicitly, rather than only implicitly in the definitons. Thank you for the suggestion.

* Page 12, Table 4.
Why is RDN_Bayes doing so poorly relative to RDN_Boost and MLN_Boost on the Mutagenesis dataset? What about Mutagenesis that is tripping up RDN_Bayes?

There is a definitional deterministic association between two relationships in Mutagenesis (bond and atom). The implementation of the LAJ algorithm we used does not learn the deterministic association, whereas the gradient boosting method does. So for the relationship predicate that can be predicted deterministically, the boosting networks shows perfect performance, which leads to a big difference in average. For the other predicates in Mutagenesis, RDN_Bayes is competitive, similar to the other datasets.

Reproducing RDN-Boost results.

Reviewer 1 raised several questions about why the numbers we reported are different from those reported in the previous gradient boosting papers. This involves many details that we address in this section of our reply.
The main difference between the experiments we report and those in the RDN-Boost paper by Natarajan et al. is that they report accuracy results for one or two target predicates, whereas we report the average accuracy over all binary predicates. Since dependency networks are a generative model that jointly models all predicates, so considering more predicates seemed to us to be an even stronger evaluation.
That said, we could of course report the results for the target predicates selected in the RDN-Boost paper separately. However, although we made a lot of effort to reproduce the results in the RDN-Boost paper by Natarajan et al. we were unsuccessful, for reasons we detail now. While we are happy to add comparisons on more datasets, the

concern of the reviewer seemed not to be that our 6 datasets were an insufficient number, but with reproducing the RDN-Boost results exactly. We detail the difficulties with reproducibility below.

Details on RDN-Boost reproducibility.

Our paper reports measurements that were obtained by using the code posted by the RDN-Boost creators, available from http://pages.cs.wisc.edu/~tushar/Boostr/down.html. We followed the instructions at http://pages.cs.wisc.edu/~tushar/Boostr/tutorial.html . On some datasets, the results of their posted code are different from what they report in the RDN-Boost paper. We contacted the RDN-Boost creators about this discrepancy but have received no reply.

Reproducibility details for each dataset. The datasets with individual prediction targets in the RDN-Boost paper are UW, MovieLens, OMOP, and WebKB.

UW. We used the data posted at http://pages.cs.wisc.edu/~tushar/Boostr/datasets/uw.zip. This includes data and a "background file" with mode declaration. The biggest difference is that we report accuracy results for all predicates in the dataset, whereas the Natarajan et al. paper reports only results for the "AdvisedBy" target. We'd be happy to report results for "AdvisedBy" separately.

However, there is still a reproducibility issue for "AdvisedBy". Running this setup produces the following measurements for the target "AdvisedBy", which is the target chosen by Natarajan et al. (Section 4.1).

AUC ROC = 0.982
AUC PR = 0.400
CLL = -0.132

[Actual output for folder 0 from RDN-Boost code]
[zqian@cs-oschulte-02 scripts]$ ./run_uw_rdn.sh
....
% Computing Area Under Curves.
%Pos=16
%Neg=2385
%LL:-6.574941231859055
%LL:-293.39050475670047

% Running command: java -jar ../auc.jar ../data/uw-cse_rdn/test0_advisedby/AUC/aucTemp.txt list 0.0
% WAITING FOR command: java -jar ../auc.jar ../data/uw-cse_rdn/test0_advisedby/AUC/aucTemp.txt list 0.0
% DONE WAITING FOR command: java -jar ../auc.jar ../data/uw-cse_rdn/test0_advisedby/AUC/aucTemp.txt list 0.0
% F1 = 1.0
% Threshold = 0.44207531332577377

% AUC ROC = 0.975708
% AUC PR = 0.351768
% CLL = -0.122195
% Precision = 0.131579 at threshold = 0.500
% Recall = 0.937500
% F1 = 0.230769

% Total inference time (20 trees): 4.432 seconds.

This is the output for folder 0, the other values are similar. The AUC-ROC is similar to that in Table 1 of Natarajan et al., but the AUC-PR is much lower: 0.36 vs. 0.95. We don't know why. As mentioned, we received no reply to a query about this from the creator of the RDN Boostr package.

MovieLens. Here too the main difference is that we report accuracy results for on all binary predicates (e.g., gender) in the dataset, whereas the Natarajan et al. paper reports only results for the Likes target. We'd be happy to report results for Likes separately. However, even so it would be difficult to reproduce the results reported in Natarajan et al, for the following reasons.

1.Natarajan et al. subsampled a set of 100 users and 603 movies randomly. This subset is not posted so we cannot get the exact same data as theirs.

2.They report using four aggregators in the paper. However, the posted RDN-code does not support the use of aggregators.

We made our own subsample of 100 users, with the following results on the same predicates as in our paper.
MovieLens (100 users, 80% training, 20% test)
AUC-PRCLL
RDN_Boost0.59-0.65
MLN_Boost0.56-1.48
RDN_Bayes1.0-0.43
[zhensong: how come AUC is not perfect? AUC is about classifier]

OMOP. As Natarajan et al. state, "we used the OMOP simulator to generate a dataset of 10,000 patients that included records of drugs and diagnoses". This random dataset is not available.

WebKB. The Boostr website posts a version of this dataset at http://pages.cs.wisc.edu/~tushar/Boostr/datasets/webkb.zip, for the target "faculty" (not "Student" as discussed in Natarajan et al. Sec. 4.3). Running this setup produces the following measurements

AUC ROC = 1
AUC PR = 1
CLL = -0.04

In other words, perfect classification performance. We looked at the rules to see how this performance is achieved. It's because URLs are partitioned into Faculty and Student URLs. When classifying a URL as Faculty or not, the facts.txt background file includes the information about the Student status. For instance, to classify

faculty(ahttpwwwcsutexaseduusersxfeng)

the program has access to the background fact that

student(ahttpwwwcsutexaseduusersxfeng)

The rules learned essentially say that if a URL is a student URL, it's not a faculty URL, and otherwise it's a Faculty URL. Here is an example of a regression tree learned.

% FOR faculty(A):
% if (student(A))
% then return -0.14185106490048832; // std dev = 0.000, 186.000 (wgt'ed) examples reached here. /* #neg=186 */
% else if (sameperson(A, A))
% | then return 0.8581489350995111; // std dev = 4.94e-08, 107.000 (wgt'ed) examples reached here. /* #pos=107 */
% | else return -0.1418510649004878; // std dev = 0.000, 17.000 (wgt'ed) examples reached here. /* #neg=17 */

We ran our RDN-Bayes system on the same data and also achieved perfect classification. With all due respect to the RDN-Boost group, this does not seem a very interesting finding. We'd be happy to add classification over all predicates in the WebKB dataset as a comparison if that strengthens the paper.

MLJ contribution information sheet (pls see the instructions)

MLJ contribution information sheet

*Each submission to the MLJ must be accompanied by a **1-2 page** information sheet that clearly answers the following questions. (Note that reviewers will expect these answers to be part of the introductory material of your paper; if this isn't currently the case, we strongly suggest you to revise your current manuscript before submission.) Please put the same care and effort in preparing this information sheet as you expect the reviewers to put in their reviews. Papers with incomplete or uninformative information sheets will be **returned without review**.*

1. What is the main claim of the paper? Why is this an important contribution to the machine learning literature?

Answer: *We present a novel approach to learning dependency networks: first learn a Bayesian network, then perform a closed-form transformation of the Bayesian network to a dependency network. Learning RDNs via BNs scales much better to large datasets than with boosting, and provides competitive accuracy in predictions.*

Many organizations store their information using relational structures (e.g., relational databases). Machine learning researchers have extensively investigated reasoning about such relational structures using formal logic. An important extension of this work is combining this formal logical reasoning with probabilistic reasoning about uncertainty, in particular to combine logic with Bayes nets, which have been widely used to represent uncertainty. We present a new method for making probabilistic predictions/inferences about relational structures using logic-based Bayes nets. We describe and evaluate fast algorithms for learning Bayes nets from relational data that perform well with our prediction method. Other researchers can use our results in two ways: (1) to perform inference for probabilistic queries about relational facts, given other facts in a relational structure. (2) To learn a Bayes net that represents statistical patterns in big relational data.

2. What is the evidence you provide to support your claim? Be precise.

Answer: *We introduce a relational adaptation of the standard BN log-linear equation for the probability of a target node conditional on an assignment of values to its Markov blanket. The new log-linear equation uses a sum of expected values of BN log-conditional probabilities, with respect to a random instantiation of first-order variables. This is equivalent to using feature instantiation proportions as feature functions. We compare our approach to state-of-the-art functional gradient boosting methods on six benchmark datasets, using metrics of speed (time to learn) and accuracy (conditional log-likelihood and area under precision-recall curve). For most data sets, our method was much faster and of comparable accuracy to the functional gradient boosting methods.*

3. What papers by other authors make the most closely related contributions, and how is your paper related to them?
Answer:

1). David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, Carl Kadie, and Pack Kaelbling. Dependency networks for inference, collaborative ltering, and data visualization. JMLR, 1:49-75, 2000.

2). Jennifer Neville and David Jensen. Relational dependency networks. JMLR, 8:653-692, 2007.

3). Sriraam Natarajan, Tushar Khot, Kristian Kersting, Bernd Gutmann, and Jude W. Shavlik. Gradient-based boosting for statistical relational learning: The relational dependency network case. Machine Learning, 86(1):25-56, 2012.

4). Tushar Khot, Sriraam Natarajan, Kristian Kersting, and Jude W. Shavlik. Learning Markov logic networks via functional gradient boosting. In ICDM, pages 320-329, 2011.

In Paper 1 the dependency networks were introduced by Heckerman et al.

In Paper 2 Neville and Jensen extend dependency networks to relational data.

In Paper 3 and Paper 4 the authors introduced the functional gradient boosting method for applying discriminative learning to build a generative graphical model (i.e. RDN, MLN).

The boosting approach to constructing a dependency network by learning a collection of discriminative models is very different from learning a Bayesian network. There are various options for hybrid approaches that combine the strengths of both. (1) Fast Bayesian network learning can be used to select features. Discriminative learning methods should work faster restricted to the BN Markov blanket of a target node. (2) The Bayesian network can provide an initial dependency network structure. Gradient boosting can then be used to fine-tune a discriminative model of a child node given parent nodes, replacing a flat conditional probability table.

4. Have you published parts of your paper before, for instance in a conference? If so, give details of your previous paper(s) and a precise statement detailing how your paper provides a significant contribution beyond the previous paper(s).

Answer: *A preliminary version of this paper was presented at the StarAI 2012 workshop. A second version of this paper was presented at ILP 2014 but not published in the conference proceedings.*

Main differences include: 1) a more extensive empirical evaluation, 2) relating our inference method to log-linear models, 3) discussion and proof of Consistency for RDNs.

Responses to ILP14 Reviews

Review1	Answers
Theoretical complexity study in section 3	Explained in Section 5.1
Pseudo code of Figure 2	Updated figure 2 and added two algorithms

Review 2	Answers
RDN consistency	Refer to Section 5.2 and the appendix for the details
Relational models v.s. Propositional models	
Why our learning method is performing so well	Answer presented in Section 6.3
only one large dataset	As indicated in Section 6.1, added even larger one IMDb

Review 3	Answers
generate DN parameters given BN parameters	Addressed by adding Algorithm 2 and elaborating Table 1.
Minor typos	All fixed

Fast Learning of Relational Dependency Networks

Oliver Schulte, Zhensong Qian, Arthur E. Kirkpatrick
Xiaoqian Yin, Yan Sun

School of Computing Science, Simon Fraser University, Canada
{oschulte,zqian,ted,xiaoqian_yin,sunyans}@sfu.ca

Abstract. A Relational Dependency Network (RDN) is a directed graphical model widely used for multi-relational data. These networks allow cyclic dependencies, necessary to represent relational autocorrelations. We describe an approach for learning both the RDN's structure and its parameters, given an input relational database: First learn a Bayesian network (BN), then transform the Bayesian network to an RDN. Thus fast Bayesian network learning translates into fast RDN learning. The BN-to-RDN transform comprises a simple, local adjustment of the Bayesian network structure and a closed-form transform of the Bayesian network parameters. This method can learn an RDN for a dataset with a million tuples in minutes. We empirically compare our approach to a state-of-the-art RDN learning approach that applies functional gradient boosting, using six benchmark datasets. Learning RDNs via BNs scales much better to large datasets than learning RDNs with current boosting methods, and provides competitive accuracy in predictions.

1 Introduction

Learning graphical models is one of the main approaches to extending machine learning for relational data. Two major classes of graphical models are dependency networks (DNs) [15] and Bayesian networks (BNs) [35]. We describe a new approach to learning dependency networks: first learn a Bayesian network, then convert that network to a dependency network. This hybrid approach combines the speed of learning Bayesian networks with the advantages of dependency network inference for relational data. Our experiments show that the hybrid learning algorithm can produce dependency networks for large and complex databases, up to one million records and 19 predicates. The predictive accuracy of the resulting networks is competitive with those from state-of-the-art function gradient boosting methods but scales substantially better than the boosting methods.

Motivation. Our approach combines the different strengths of Bayesian networks and dependency networks for relational learning. The special strength of Bayesian networks is scalability in learning [34, Sec.8.5.1],[21]. Bayesian networks offer closed-form parameter estimation via the maximum likelihood method, and

therefore closed-form model evaluation. Model evaluation is the computationally most expensive part of relational learning, as it requires combining information from different related tables, which involves expensive table joins.

In contrast, a strength of relational dependency networks is that they support inference in the presence of cyclic dependencies [34, 32]. Cyclic dependencies occur when a relational dataset features auto-correlations, where the value of an attribute for an individual depends on the values of the same attribute for related individuals. Figure 1 below provides an example. It is difficult for Bayesian networks to model auto-correlations because by definition, the graph structure of a Bayesian network must be acyclic [7, 43, 13]. Because of the importance of relational auto-correlations, dependency networks have gained popularity since they support reasoning about cyclic dependencies using a directed graphical model.

These advantages of our hybrid approach are specific to relational data. For propositional (i.i.d.) data, which can be represented in a single table, there is no problem with cyclic dependencies, and the acyclic constraint of Bayesian networks can actually make probabilistic inference more efficient [17]. Also, the closed-form model evaluation of Bayesian networks is relatively less important in the single-table case, because iterating over the rows of a single data table to evaluate a dependency network is relatively fast compared to iterating over ground atoms in the relational case, where they are stored across several tables.

Our approach extends the moralization approach to relational structure learning, which learns an undirected Markov model by first learning a Bayesian network structure, then converts the network structure to a Markov logic structure, without parameters [21, 39]. That approach also combines the scalable learning of Bayesian networks with the support undirected models offer for inference with auto-correlation. Our present work extends this by also computing the dependency network parameters from the learned Bayesian network. The previous work used only the Bayesian network structure. Our theoretical analysis shows that the learned dependency networks provide different predictions from Markov networks.

Contributions. We make three main contributions:

1. A faster approach for learning relational dependency networks: first learn a Bayesian network, then convert it to a dependency network.
2. A closed-form log-linear discriminative model for computing the relational dependency network parameters from Bayesian network structure and parameters.
3. Necessary and sufficient conditions for the resulting network to be **consistent**, defined as the existence of a single joint distribution that induces all the conditional distributions defined by the dependency network [15].

2 Bayesian Networks and Relational Dependency Networks

We review dependency networks and their advantages for modelling relational data. We assume familiarity with the basic concepts of Bayesian networks [35].

2.1 Dependency networks and Bayesian networks

The structures of both Bayesian networks and dependency networks are defined by a directed graph whose nodes are random variables. Bayesian networks must be acyclic, while dependency networks may contain cycles, including the special case of bi-directed edges. For both networks, the parameters are conditional distributions over the value of a node given its parents. The two types differ in the influence of a node's children, however. In a Bayesian network, a node is only independent of all other nodes given an assignment of values to its parents, its children, and the co-parents of its children, whereas in a dependency network a node is independent given an assignment of values to only its parents. In graphical model terms, the **Markov blanket** of a node in a dependency network, the minimal set of nodes such that assigning them values will make this node independent of the rest of the network, is simply its parents.[1] For a Bayesian network, the Markov blanket is the node's parents, children, and the co-parents of its children.

Consequently, a conditional probability in a dependency network effectively specifies the probability of a node value given an assignment of values to *all* other nodes. Following Heckerman *et al.* [15], we refer to such conditional probabilities as **local probability distributions**.

2.2 Relational Dependency Networks

Relational dependency networks [34] extend dependency networks to model distributions over multiple populations. We present the relational case using the parametrized random variable notation [24]. A functor is a symbol denoting a function or predicate. Each functor has a set of values (constants) called the **domain** of the functor. Functors with boolean ranges are called **predicates** and their name is capitalized. We consider only functors with finite domains. An expression $f(\tau_1, \ldots, \tau_k)$, where f is a functor and each τ_i is a first-order variable or a constant, is a **Parametrized Random Variable** (PRV). A directed acyclic graph whose nodes are PRVs is a **parametrized Bayesian network structure**, while a general (potentially cyclic) directed graph whose nodes are PRVs is a **relational dependency network structure** (RDN). A Bayesian network structure or relational dependency network structure augmented with the appropriate conditional probabilities is respectively a **Bayesian network template** or **relational dependency network template**. Note that the RDN templates that we define in this paper have the same Markov blanket as the Bayesian network templates from which they are derived but a different edge structure and probabilities. Algorithm 1, defined in Section 3, converts the probabilities of the Bayesian template to their counterparts in the relational dependency template.

RDNs extend dependency networks from i.i.d. to relational data via knowledge-based model construction [34]: The first-order variables in a template RDN graph

[1] For this reason Hofmann and Tresp originally used the term "Markov blanket networks" for dependency networks [16].

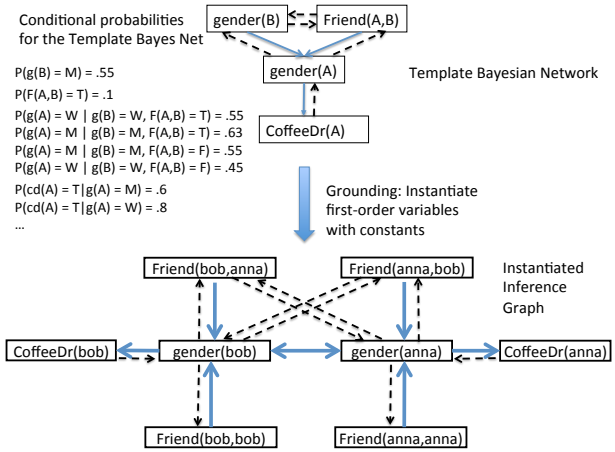

Fig. 1. A Bayesian/dependency template network (top) and the instantiated inference graphs (bottom). By convention, predicates (Boolean functors) are capitalized. Edges from the BN template are solid blue, while edges added by the BN-to-DN transformation are dashed black. The edge set in the DN comprises both solid and dashed arrows. Note that although the template BN (top) is acyclic, its instantiation (bottom) features a bi-directed edge between $gender(bob)$ and $gender(anna)$.

are instantiated for a specific domain of individuals to produce an *instantiated* or *ground* propositional DN graph, the **inference graph**. Figure 1 gives a dependency network template and its inference graph. Given an edge in the template RDN, instantiating both the parent and the child of the edge with the same grounding produces an edge in the inference graph. An example local probability distribution for the graph in Figure 1 (abbreviating functors) is

$$P(g(anna)|g(bob), CD(anna), F(anna, bob), F(bob, anna), F(anna, anna)).$$

Language Bias. The general definition of a parametrized random variable allows PRVs to contain constants as well as population variables. Another language extension is to allow parametrized random variables to be formed with aggregate functions, as described by Kersting and deRaedt [19]. For example, it is possible to use a functor that returns the number of friends of a generic person $\mathbb{A}$. The main contribution of this paper, our relational BN-to-DN conversion method, can be used whether the parametrized random variables contain constants, aggregates, or only first-order variables. A common restriction to simplify model structure learning is to exclude constants (e.g. [9, 6]). Since this restriction applies only to the previous structure learning methods that we apply in this paper, we discuss it further in the Bayesian network learning section 6 below. Friedman *et al.* investigated learning directed graphical models with aggregate functions [9].

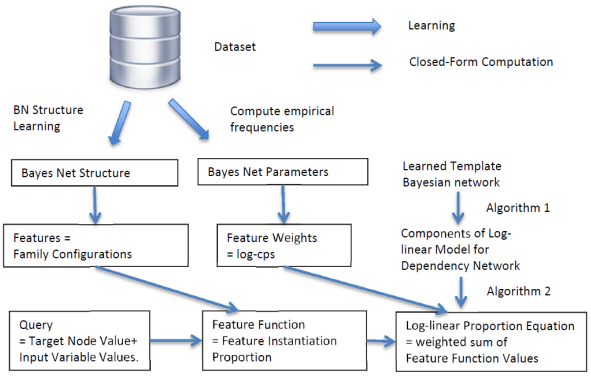

Fig. 2. The program flow for computing local probability distributions from a template Bayesian network. Features and weights are computed from the Bayesian network. Feature function values are computed for each query.

3 Learning Relational Dependency Networks via Bayesian Networks

Our algorithm for rapidly learning relational dependency networks (Figure 2) begins with any relational learning algorithm for Bayesian networks. Using the resulting Bayesian network as a template, we then apply a simple, fast transformation to obtain a relational dependency template. Finally we apply a closed-form computation to derive the dependency network inference graph parameters from the Bayesian structure and parameters.

BN-to-DN structure conversion. Converting a Bayesian network structure to a dependency network structure is simple: for each node, add an edge pointing to the node from each member of its BN Markov blanket [15]. The result contains bidirectional links between each node, its children, and its co-parents (nodes that share a child with this one). This is equivalent to the standard moralization method for converting a BN to an undirected model [6], except that the dependency network contains bi-directed edges instead of undirected edges. Bidirected edges have the advantage that they permit assignment of different parameters to each direction, whereas undirected edges have only one parameter.

BN-to-DN parameter conversion. For propositional data, converting Bayesian network parameters to dependency network parameters is simple: apply the standard BN product formula and solve for the local probability distributions given Bayesian network parameters [37, Ch.14.5.2]. A **family** comprises a node and its parents. A **family configuration** specifies a value for a child node and each of its parents. For example, in the template of Figure 1 (top), one family is

$gender(\mathbb{A}), Friend(\mathbb{A}, \mathbb{B}), gender(\mathbb{B})$ and one of its eight possible configurations is

$$gender(\mathbb{A}) = \mathrm{M}, Friend(\mathbb{A}, \mathbb{B}) = \mathrm{T}, gender(\mathbb{B}) = \mathrm{M}.$$

The Markov blanket of a target node comprises multiple families, one each for the target node and each of its children, so an assignment of values to the target's Markov blanket defines a unique configuration for each family. Hence in the propositional case the Markov blanket induces a *unique* log-conditional probability for each family configuration. The probability of a target node value given an assignment of values to the Markov blanket is then proportional to *the exponentiated sum of these log-conditional probabilites* [37, Ch.14.5.2].

With relational data, however, different family configurations can be simultaneously instantiated, *multiple times.* We generalize the propositional log-linear equation for relational data by replacing the unique log-conditional probability with the *expected* log-conditional probability that results from selecting an instantiation of the family configuration uniformly at random. The probability of a target node value given an assignment of values to the Markov blanket is then proportional to the exponentiated sum of the expected log-conditional probabilites. We describe the resulting closed-form equation in the next section.

Algorithm 1: Computing Features and Weights for Template Dependency Network.

Input: Template Bayesian Network B (Structure and Parameters)
Output: A List of Relevant Features; a Weight for each Feature

for each target node T **do**
  initialize $Feature\_Weight\_List(T)$ as the empty list
  for each U in $\{T \cup \mathrm{Ch}(T)\}$ **do**
    for each value u of the child node U **do**
      for each vector of parent values $\boldsymbol{u}_{pa}$ **do**
        Feature $F := (U = u, \mathrm{Pa}(U) = \boldsymbol{u}_{pa})$
        FeatureWeight $w := \ln \theta(U = u | \mathrm{Pa}(U) = \boldsymbol{u}_{pa})$
        if the Feature F does not contain a false relationship other than T **then**
          add (F, w) to $Feature\_Weight\_List(T)$
        end if
      end for
    end for
  end for
end for
return $Feature\_Weight\_List(T)$

Algorithm 2: Computing local probability distributions, the parameters of the Inference Dependency Network.

Input: Feature-Weight List of Dependency Network, Query $P(T^* = t|\Lambda^*) =?$. T is a template node, $T^* = T\gamma$ is the target grounding.
Output: Normalized log-linear score

initialize $score(T^* = t) := 0$
for each Feature $F = (U = u, \mathrm{Pa}(U) = \boldsymbol{u}_{pa})$ in $Feature\_Weight\_List(T)$ **do**
  Let w be the weight listed for feature F
  {Next compute feature function.}
  $RelFamCnt(F) := \mathrm{n^r}\left[\gamma; \mathsf{U} = u, \mathrm{Pa}(\mathsf{U}) = \boldsymbol{u}_{pa}; T^* = t, \Lambda^*\right]$
  $TotalRelFamCnt(U) := \sum_{u', \boldsymbol{u}'_{pa}} \mathrm{n^r}\left[\gamma; \mathsf{U} = u', \mathrm{Pa}(\mathsf{U}) = \boldsymbol{u}'_{pa}; T^* = t, \Lambda^*\right]$
  $FamilyProportion\ \mathrm{p^r}(F) := RelFamCnt(F)/\,TotalRelFamCnt(U)$
  $score(T^* = t)\ \mathrel{+}= \mathrm{p^r} \cdot w$
end for
return *Normalized scores for target node.*

4 The Log-linear Proportion Equation

We propose a log-linear equation, the **log-linear proportion equation** (lower right box of Figure 2), for computing a local probability distribution for a ground target node, T^*, given (i) a target value t for the target node, (ii) a complete set of values Λ^* for all ground terms other than the target node, and (iii) a template Bayesian network. The template structure is represented by functions that return the set of parent nodes of U, $\mathrm{Pa}(U)$, and the set of child nodes of U, $\mathrm{Ch}(U)$. The parameters of the template are represented by the conditional probabilities of a node U having a value u conditional on the values of its parents, $\theta(U = u|\mathrm{Pa}(U) = \boldsymbol{u}_{pa})$. A grounding γ substitutes a constant for each member of a list of first-order variables, $\{\mathbb{A}_1 = a_1, \ldots, \mathbb{A}_k = a_k\}$. Applying a grounding to a template node defines a fully ground target node: $gender(\mathbb{A})\{\mathbb{A} = sam\} = gender(sam)$. These are combined in the following log-linear equation to produce a local probability distribution:

Definition 1 (The Log-Linear Proportion Equation).

$$P(T^* = t|\Lambda^*) \propto \exp \sum_{U} \sum_{u, \boldsymbol{u}_{pa}} \left[\ln \theta(U = u|\mathrm{Pa}(U) = \boldsymbol{u}_{pa})\right] \cdot \mathrm{p^r}\left[\gamma; U = u, \mathrm{Pa}(U) = \boldsymbol{u}_{pa}; T^* = t, \Lambda^*\right]$$

where

U varies over $\{T\} \cup \mathrm{Ch}(T)$;
the singleton value u varies over *the range of* U;
the vector of values $\boldsymbol{u}_{pa}$ varies over *the product of the ranges of* U*'s parents, constrained to value* t *for occurrences of* T;
$T^* = T\gamma$ is *is the target grounding of template node* T;
and $\mathrm{p^r}$ is *the feature function, the family proportion.*

The family proportion p^{r} is computed as follows:

1. For a given family configuration $(U = u, \mathrm{Pa}(U) = \boldsymbol{u}_{pa})$, let the **family count**
$$\mathrm{n}\left[\gamma; U = u, \mathrm{Pa}(U) = \boldsymbol{u}_{pa}; T^* = t, \Lambda^*\right]$$
be the number of instantiations that (a) satisfy the family configuration and the ground node values specified by $T^* = t, \Lambda^*$, and (b) are consistent with the equality constraint defined by the grounding γ.
2. The **relevant family count** n^r is 0 if the family configuration contains a false relationship (other than the target node), else equals the family count. It is common in statistical-relational models to restrict predictors to existing relationships only [11, 37].
3. The **family proportion** is the relevant family count, divided by the total sum of all relevant family counts for the given family:
$$\mathrm{p}^{\mathrm{r}}\left[\gamma; U = u, \mathrm{Pa}(U) = \boldsymbol{u}_{pa}; T^* = t, \Lambda^*\right] = \frac{\mathrm{n}^{\mathrm{r}}\left[\gamma; U = u, \mathrm{Pa}(U) = \boldsymbol{u}_{pa}; T^* = t, \Lambda^*\right]}{\sum_{u', \boldsymbol{u}'_{pa}} \mathrm{n}^{\mathrm{r}}\left[\gamma; U = u', \mathrm{Pa}(U) = \boldsymbol{u}'_{pa}; T^* = t, \Lambda^*\right]}$$

In our experiments, family counts and proportions are computed using exact counting methods (see Section 4.3 below).

4.1 Example and Pseudocode

Table 1 illustrates the computation of these quantities for predicting the gender of a new test instance (*sam*). Algorithm 1 shows pseudocode for the closed-form transformation of Bayesian network structure and parameters into features and weights for the dependency network. Algorithm 2 shows pseudocode for computing the scores defined by the log-linear Equation (1), given a list of weighted features and a target query.

The inner sum of Equation (1) computes the expected log-conditional probability for a family with child node U, when we randomly select a relevant grounding of the first-order variables in the family.

4.2 Discussion and Motivation

We discuss the key properties of our local distribution model, Equation (1).

Log-Linearity. The survey by Kimmig *et al.* [24] shows that most statistical-relational methods define log-linear models. The general form of a discriminative log-linear model [42] is that the conditional probability of a target variable value given input variable values is proportional to an exponentiated weighted sum of feature functions. A feature function maps a complete assignment of ground node values (= target value + input variables) to a real number. Khazemi *et al.* have shown that many relational aggregators can be represented by a log-linear model with suitable features [18]. Equation (1) instantiates this well-established log-linear schema as follows: The features of the model are the family configurations

Table 1. Applying the log-linear proportion equation with the Bayesian network of Figure 1 to compute $P(gender(sam) = \mathrm{W}|\Lambda^*)$ and $P(gender(sam) = \mathrm{M}|\Lambda^*)$. Each row represents a feature/family configuration. For the sake of the example we suppose that the conjunction Λ^* specifies that Sam is a coffee drinker, has 60 male friends, and 40 female friends. CP is the conditional probability BN parameter of Figure 1 and $w \equiv \ln(CP)$.

Child Value u	Parent State $\boldsymbol{u}_{pa}$	CP	w	p^{r}	$w \times \mathrm{p}^{\mathrm{r}}$
$g(sam) = \mathrm{W}$	$g(B) = \mathrm{W}$, $F(sam, B) = \mathrm{T}$	0.55	−0.60	0.4	−0.24
$g(sam) = \mathrm{W}$	$g(B) = \mathrm{M}$, $F(sam, B) = \mathrm{T}$	0.37	−0.99	0.6	−0.60
$CD(sam) = \mathrm{T}$	$g(sam) = \mathrm{W}$	0.80	−0.22	1.0	−0.22
$CD(sam) = \mathrm{F}$	$g(sam) = \mathrm{W}$	0.20	−1.61	0.0	0.00
Sum $(\exp(Sum) \propto P(gender(sam) = \mathrm{W}\|\Lambda^*))$					−1.06
$g(sam) = \mathrm{M}$	$g(B) = \mathrm{W}$, $F(sam, B) = \mathrm{T}$	0.45	−0.80	0.4	−0.32
$g(sam) = \mathrm{M}$	$g(B) = \mathrm{M}$, $F(sam, B) = \mathrm{T}$	0.63	−0.46	0.6	−0.28
$CD(sam) = \mathrm{T}$	$g(sam) = \mathrm{M}$	0.60	−0.51	1.0	−0.51
$CD(sam) = \mathrm{F}$	$g(sam) = \mathrm{M}$	0.40	−0.92	0.0	0.00
Sum $(\exp(Sum) \propto P(gender(sam) = \mathrm{M}\|\Lambda^*))$					−1.11

$(U = u, \mathrm{Pa}(U) = \boldsymbol{u}_{pa})$ where the child node is either the target node or one of its children. The feature weights are the log-conditional BN probabilities defined for the family configuration. The input variables are the values specified for the ground (non-target) nodes by the conjunction Λ^*. The feature functions are the family proportion p^{r}. Like other log-linear relational models, Equation 1 enforces parameter tying, where different groundings of the same family configuration receive the same weight [24].

Standardization. Using proportions as feature functions has the desirable consequence that the range of all feature functions is standardized to [0,1]. It is well-known that the number of instantiation counts in relational data can differ for different families, depending on the population variables they contain. This ill-conditioning causes difficulties for log-linear models because families with more population variables can have an exponentially higher impact on the score prediction [30]. Intuitively, counts tacitly conflate number of instantiations with degree of information. Proportions avoid such ill-conditioning.

Generalizing the Propositional Case. A useful general design principle is that relational learning should have propositional learning as a special case [28, 25]: When we apply a relational model to a single i.i.d. data table, it should give the same result as the propositional model. Equation 1 satisfies this principle. In the propositional case, an assignment of values to all nodes other than the target

node specifies a *unique* value for each family configuration. This means that all the family counts n^r are either 0 or 1, hence all relevant proportions p^r are 0 or 1, depending on whether a family configuration matches the query or not. For a simple illustration, consider the edge $gender(\mathbb{A}) \rightarrow CoffeeDr(\mathbb{A})$. Since this edge concerns only the *Person* domain associated with the single population variable $\mathbb{A}$, we may view this edge as a propositional subnetwork. Suppose the query is $P(gender(sam) = \mathrm{W}|CoffeeDr(sam) = \mathrm{T})$. The only family configurations with nonzero counts are $gender(sam) = \mathrm{W}$ (count 1) and $CoffeeDr(sam) = \mathrm{T}), gender(sam) = \mathrm{W}$ (count 1). Equation (1) gives

$$P(g(sam) = \mathrm{W}|CD(sam) = \mathrm{T}) \propto$$
$$\exp\{\ln P(g(sam) = \mathrm{W}) + \ln P(CD(sam) = \mathrm{T})|g(sam) = \mathrm{W})\}.$$

This agrees with the propositional BN formula for a local conditional probability, which is the product of the BN conditional probabilities for the target node given its children, and the target node's children given their parents. This formula can be derived from the BN product rule for defining a joint probability [37, Ch.14.5.2]. In our simple two-node example, it can be derived immediately from Bayes' theorem:

$$P(g(sam) = \mathrm{W}|CD(sam) = \mathrm{T}) \propto$$
$$P(CD(sam) = \mathrm{T})|g(sam) = \mathrm{W}) \times P(g(sam) = \mathrm{W}),$$

which agrees with the solution above derived from Equation (1). It may seem surprising that in predicting gender given coffee drinking, the model should use the conditional probability of coffee drinking given gender. However, Bayes' theorem states that $P(X|Y)$ is proportional to $P(Y|X)$. In our example, given that the BN model specifies that women are more likely to be coffee drinkers than men, the information that Sam is a coffee drinker raises the probability that Sam is a woman.

4.3 Complexity of Algorithms 1 and 2

The loops of Algorithm 1 enumerate every family configuration in the template Bayesian network exactly once. Therefore *computing features and weights takes time linear in the number of parameters of the Bayesian network.*

Evaluating the log-linear equation, as shown in Algorithm 2, requires finding the number of instantiations that satisfy a conjunctive family formula, given a grounding. This is an instance of the general problem of computing the number of instantiations of a formula in a relational structure. Computing this number is a well-studied problem with highly efficient solutions [45, 40].

A key parameter is the number m of first-order variables that appear in the formula. A loose upper bound on the complexity of counting instantiations is d^m, where d is the maximum size of the domain of the first-order variables. Thus

counting instantiations has parametrized polynomial complexity [8], meaning that if m is held constant, counting instantiations requires polynomially many operations in the size of the relational structure (i.e., the size of $T^* = t, \Lambda^*$ in Equation (1)). For varying m, the problem of computing the number of formula instantiations is #P-complete [7, Prop.12.4].

5 Consistency of the Derived Dependency Networks

A basic question in the theory of dependency networks is the *consistency* of the local probabilities. Consistent local probabilities ensure the existence of a single joint probability distribution p that induces the various local conditional probability distributions P for each node

$$P(T^* = t|\Lambda^*) \propto p(T^* = t, \Lambda^*)$$

for all target nodes T^* and query conjunctions Λ^* [15].

We present a precise condition on a template Bayesian network for its resulting dependency network to be consistent and the implications of those conditions. We define an **edge-consistent template Bayesian network** to be a network for which every edge has the same set of population variables on both nodes.

Theorem 1. *A template Bayesian network is edge-consistent if and only if its derived dependency network is consistent.*

The proof of this result is complex, so we present it in an appendix. Intuitively, in a joint distribution, the correlation or potential of an edge is a single fixed quantity, whereas in Equation (1), the correlation is adjusted by the size of the relational neighbourhood of the target node, which may be either the child or the parent of the edge. If the relational neighborhood size of the parent node is different from that of the child node, the adjustment makes the conditional distribution of the child node inconsistent with that of the parent node. The edge-consistency characterization shows that the inconsistency phenomenon is properly relational, meaning it arises when network structure contains edges that relate parents and children from different populations.

The edge-consistency condition required by this theorem is quite restrictive: Very few template Bayesian networks will have exactly the same set of population variables on both sides of each edge.[2] Therefore relational template Bayesian networks, which have multiple population variables, will most often produce inconsistent dependency networks. Previous work has shown that dependency networks learned from data are almost always inconsistent but nonetheless provide accurate predictions using ordered pseudo-Gibbs sampling [15, 34, 29] or Generative Stochastic Networks [2, Sec.3.4].

[2] A commonly used weaker condition is range-restriction: that the population variables in the child node should be contained in the population variables of its parents [19], but not vice versa as with edge-consistency.

6 Bayesian Network Learning

Given a network structure, the BN parameters can be estimated by applying the maximum likelihood principle, using the conditional frequencies observed in a relational database. These were computed using previously-published algorithms for multi-relational data [40]. For completeness, we briefly describe the learn-and-join method. We also review some of the fundamental insights and results from previous work concerning the scalability of Bayesian network learning, for both propositional and relational data. Readers who are mainly interested in our empirical findings can skip this section without loss of continuity.

For structure learning, we used the learn-and-join (LAJ) algorithm [21]. This is a state-of-the-art Bayesian network structure learning algorithm with an iterative deepening search strategy similar to that introduced by Friedman *et al.* [9]. We used the implementation by the creators of the LAJ algorithm, which is available on-line [20]. It is important to note that this implementation incorporates a language bias: it considers only parametrized random variables without any constants, population variables only. This is a common restriction for statistical-relational structure learning methods (e.g. [9, 6]), which trades off expressive power for faster learning. The boosting systems for learning dependency networks do not impose this restriction and search for rules that may contain both first-order variables and constants. In our experiments, the predictive accuracy of the Bayesian network approach was good despite the restricted model language. An extension of the LAJ algorithm to include parametrized random variables with constants and/or aggregate functions (cf. Section 2.2) should improve predictive accuracy even further. We discuss in Section 9 how the boosting methods can be combined with Bayesian network learning to expand the space of logical patterns searched by the model.

6.1 The Learn-and-Join Lattice Search

We briefly review the main ideas behind the learn-and-join algorithm; for more details and examples please see [39]. The learn-and-join algorithm upgrades a single-table propositional BN learner for relational learning. The key idea of the algorithm can be explained in terms of the *table join lattice.* Recall that the (natural) join of two or more tables, written $T_1 \bowtie T_2 \cdots \bowtie T_k$ is a new table that contains the rows in the Cartesian products of the tables whose values match on common fields. A table join corresponds to logical conjunction [44]. Say that a join table J is a **subjoin** of another join table J' if $J' = J \bowtie J^*$ for some join table J^*. If J is a subjoin of J', then the fields (columns) of J are a subset of those in J'. The subjoin relation defines the table join lattice. The basic idea of the learn-and-join algorithm is that join tables should inherit edges between descriptive attributes from their subjoins. This gives rise to the following constraints for two attributes X_1, X_2 that are both contained in some subjoin of J. (i) X_1 and X_2 are adjacent in a BN B_J for J if and only if they are adjacent in a BN for some subjoin of J. (ii) if all subjoin BNs of J orient the link as $X_1 \rightarrow X_2$ resp. $X_1 \leftarrow X_2$, then B_J orients the link as $X_1 \rightarrow X_2$

resp. $X_1 \leftarrow X_2$. The learn-and-join algorithm then builds a PBN for the entire database $\mathcal{D}$ by level-wise search through the table join lattice. The user chooses a single-table BN learner. The learner is applied to table joins of size 1, that is, regular data tables. Then the learner is applied to table joins of size $s, s+1, \ldots$, where the constraints (i) and (ii) are propagated from smaller joins to larger joins.

6.2 Computational Cost and Scalability

The computational cost of Bayesian network learning has been analyzed in previous work for both propositional and relational data. We review some of the main points as they pertain to the scalability of our overall BN-to-DN conversion approach. The complexity analysis of the learn-and-join algorithm [39] shows that the key computational cost is the run-time of the propositional BN learner that is used as a subroutine in the algorithm. This cost in turn depends on two key factors: (1) Model search cost, the number of candidate models generated. (2) Model evaluation cost, the cost of evaluating a candidate model. The model evaluation cost is dominated by data access [31], especially the cost of computing sufficient statistics. In relational data, this is the cost of finding the number of groundings that satisfy a formula in the input data [9, 6, 38]. We discuss each factor in turn.

Model Search. While finding a Bayesian network that optimizes a model selection score is NP-hard [4], highly efficient local search methods have been developed that provide good approximations fast. The implementation of the LAJ algorithm that we used employs GES (Greedy Equivalence Search) as its propositional BN learning method. This search strategy has the remarkable property that in the sample size limit, it is guaranteed to find an optimal Bayesian network [5]. Thus as the sample size increases, the quality of the BN models discovered by GES increases as well, despite the NP-hardness of finding an optimal model. Given the language restriction of using only first-order variables and excluding constants, relational BN model search generates a number of models comparable to propositional model search [39]. Efficient relational BN model search with both constants and variables is to our knowledge an open problem. Our Bayes net-to-dependency net conversion does not depend on the language bias and therefore could leverage more expressive Bayesian network models to produce more expressive dependency network models.

Model Evaluation. Using the maximum likelihood scoring method, or other related scores, the fit of a Bayesian network to the input relational dataset can be evaluated in closed form given the sufficient statistics of the network [9, 38]. We discussed the complexity of this problem in Section 4.3 above. While relational counting is not an easy problem, researchers have developed efficient solutions. The important point for our experiments is that model evaluation by counting, which is most of what Bayes nets require, is much faster than iteratively computing model predictions for the ground facts in the input database, which is

the standard model evaluation approach for other graphical model classes [34, Sec.8.5.1].

7 Empirical Evaluation: Design and Datasets

There is no obvious baseline method for our RDN learning method because ours is the first work that uses the approach of learning an RDN via a Bayesian network. Instead we benchmark against the performance of our method a different approach for learning RDNs, which uses an ensemble learning approach based on functional gradient boosting. Boosted functional gradient methods have been shown to outperform previous methods for learning relational dependency networks [22, 32]. Our argument is not that the Bayes net approach is superior to boosting on all or even most datasets. The two approaches are very different, with different strengths and limitations. Our view is that ultimately the best system for learning RDNs is one that combines the strengths of Bayesian network learning with those of boosting. After we present our experimental results in detail, we make suggestions for how this combination can be achieved. Despite the fundamental differences in approach, we believe that our empirical comparison provides enough evidence that Bayesian network learning brings substantial scalability advantages, while at the same the predictive accuracy of the learned dependency networks is competitive with that of the boosting networks. Therefore the advantages in scalability and interpretability establish Bayesian network conversion as a worthwhile option for learning relational dependency networks.

All experiments were done on a machine with 8 GB of RAM and a single Intel Core 2 QUAD Processor Q6700 with a clock speed of 2.66 GHz (there is no hyper-threading on this chip), running Linux Centos 2.6.32. Code was written in Java, JRE 1.7.0. All code and datasets are available [20].

Datasets We used six benchmark real-world databases. For more details please see the references in [39]. Summary statistics are given in Table 2.

UW-CSE This dataset [26] lists facts about the Department of Computer Science and Engineering at the University of Washington, such as entities (e.g., *Person*, *Course*) and their relationships (e.g., *AdvisedBy*).[3] The experiments reported in [32] used the same dataset; their version is available in WILL format.[4]

MovieLens MovieLens is a commonly-used rating dataset.[5] It contains two entity sets, Users and Movies. For each user and movie that appears in the database, all available ratings are included. MovieLens(1M) contains 1 M ratings, 3,883 Movies, and 6,039 Users. MovieLens(0.1M) contains about 0.1 M ratings, 1,682 Movies, and 941 Users. We did not use the binary genre predicates because they are easily learned with exclusion rules.

[3] http://alchemy.cs.washington.edu/data/uw-cse/
[4] http://pages.cs.wisc.edu/ tushar/Boostr/datasets/uw.zip
[5] www.grouplens.org

Mutagenesis This dataset is widely used in inductive logic programming research. It contains information on Atoms, Molecules, and Bonds between them. We use the discretization of Schulte and Khosravi [39].

Hepatitis This data is a modified version of the PKDD02 Discovery Challenge database. The database contains information on the laboratory examinations of hepatitis B and C infected patients.

Mondial Data from multiple geographical Web data sources.

IMDb The largest dataset in terms of number of total tuples (more than 1.3M) and schema complexity. It combines MovieLens with data from the Internet Movie Database (IMDb)[6] [36].

Methods Compared Functional gradient boosting is a state-of-the-art method for applying discriminative learning to build a generative graphical model. The local discriminative models are ensembles of relational regression trees [22]. Functional gradient boosting for relational data is implemented in the Boostr system [23]. For functors with more than two possible values, we followed [22] and converted each such functor to a set of binary predicates by introducing a predicate for each possible value. We compared the following methods:

RDN_Bayes Our method: Learn a Bayesian network, then convert it to a relational dependency network.

RDN_Boost The RDN learning mode of the Boostr system [32].

MLN_Boost The MLN learning mode of the Boostr system. It takes a list of target predicates for analysis. We provide each binary predicate in turn as a single target predicate, which amounts to using MLN learning to construct an RDN. This RDN uses a log-linear model for local probability distributions that is derived from Markov Logic Networks.

We used the default Boostr settings. We experimented with alternative settings but they did not improve the performance of the boosting methods.

Prediction Metrics We follow [22] and evaluate the algorithms using conditional log likelihood (CLL) and area under the precision-recall curve (AUC-PR). AUC-PR is appropriate when the target predicates features a skewed distribution as is typically the case with relationship predicates. For each fact $T^* = t$ in the test dataset, we evaluate the accuracy of the predicted local probability $P(T^* = t|\Lambda^*)$, where Λ^* is a complete conjunction for all ground terms other than T^*. Thus Λ^* represents the values of the input variables as specified by the test dataset. CLL is the average of the logarithm of the local probability for each ground truth fact in the test dataset, averaged over all test predicates. For the gradient boosting method, we used the AUC-PR and likelihood scoring routines included in Boostr.

Both metrics are reported as means and standard deviations over all binary predicates. The learning methods were evaluated using 5-fold cross-validation.

[6] www.imdb.com, July 2013

Table 2. Learning Time. The total learning time for constructing a relational dependency network from an input database. Only partial boosting learning times are reported for the larger databases MovieLens(1M) and IMDb—see text for details. Spread is reported as coefficient of variation (CV—standard deviation / mean). PRV = Parametrized Random Variable.

			RDN_Bayes		RDN_Boost		MLN_Boost	
Dataset	kTuple	PRVs	(s)	CV	(s)	CV	(s)	CV
UW	0.6	14	**14**	0.00	237	0.06	329	0.16
Mondial	0.9	18	1836	0.07	**369**	0.06	717	0.05
Hepatitis	11.3	19	5434	0.01	6648	0.02	**3197**	0.04
Mutagenesis	24.3	11	**11**	0.00	1342	0.04	1040	0.02
MovieLens(0.1M)	83.4	7	**8**	0.07	3019	0.04	3292	0.01
MovieLens(1M)	1010.1	7/6	**8**	0.09	32230	0.04	25528	0.04
IMDb	15538.4	17/13	**9346**	0.22	78129	0.04	29704	0.03

Each database was split into 5 folds by randomly selecting entities from each entity table, and restricting the relationship tuples in each fold to those involving only the selected entities (i.e., subgraph sampling [39]). The models were trained on 4 of the 5 folds, then tested on the remaining one.

8 Results

We report learning times and accuracy metrics. In addition to these quantitative assessments, we inspect the learned models to compare the model structures. Finally we make suggestions for combining the strenghts of boosting with the strengths of Bayesian network learning.

8.1 Learning Times.

Table 2 shows learning times for the methods. The Bayesian network learning simultaneously learns a joint model for all parametrized random variables (PRVs). Recall that Boolean PRVs are predicates. For the boosting method, we added together the learning times for each target PRV. On MovieLens(1M), the boosting methods take over 2 days to learn a classifier for the relationship B_U2Base, so we do not include learning time for this predicate for any boosting method. On the largest database, IMDb, the boosting methods cannot learn a local distribution model for the three relationship predicates with our system resources, so we only report learning time for descriptive attributes by the boosting methods. Likewise, our accuracy results in Tables 3 and 4 include only measurements for descriptive attributes on the datasets IMDb and MovieLens(1M).

Consistent with other previous experiments on Bayesian network learning with relational data [21, 39], Table 2 shows that RDN_Bayes scales very well with the number of data tuples: even the MovieLens dataset with 1 M records can be analyzed in seconds. RDN_Bayes is less scalable with the number of PRVs, since it learns a joint model over all PRVs simultaneously, although the time

Table 3. Conditional Log-Likelihood: Mean (top), Std. Dev. (bottom)

					MovieLens		
Method	UW	Mond.	Hepa.	Muta.	(0.1M)	(1M)	IMDb
RDN_Boost	-0.30	-0.48	-0.48	-0.36	-0.50	**-0.22**	**-0.49**
MLN_Boost	-0.14	-0.40	-0.49	-0.23	-0.50	-0.23	**-0.49**
RDN_Bayes	**-0.01**	**-0.25**	**-0.39**	**-0.22**	**-0.30**	-0.28	-0.51
RDN_Boost	0.02	0.03	0.01	0.02	0.01	0.00	0.00
MLN_Boost	0.01	0.05	0.01	0.02	0.01	0.00	0.00
RDN_Bayes	0.00	0.06	0.10	0.07	0.00	0.00	0.00

Table 4. Area Under Precision-Recall Curve: Mean (top), Std. Dev. (bottom).

					MovieLens		
Method	UW	Mond.	Hepa.	Muta.	(0.1M)	(1M)	IMDb
RDN_Boost	0.42	0.27	0.55	0.71	0.50	0.88	0.63
MLN_Boost	0.68	0.44	0.55	**0.86**	0.50	0.88	0.63
RDN_Bayes	**0.89**	**0.79**	0.55	0.50	**0.65**	**1.00**	**0.85**
RDN_Boost	0.00	0.00	0.01	0.02	0.01	0.00	0.01
MLN_Boost	0.01	0.04	0.01	0.04	0.01	0.00	0.01
RDN_Bayes	0.00	0.07	0.11	0.10	0.02	0.00	0.00

remains feasible (1–3 hours for 17–19 predicates; see also [39]). By contrast, the boosting methods scale well with the number of predicates, which is consistent with findings from propositional learning [15]. Gradient boosting scales much worse with the number of data tuples.

8.2 Accuracy

Whereas learning times were evaluated on all PRVs, unless otherwise noted, we evaluate accuracy on all the binary predicates only (e.g., *gender*, *Borders*) because the boosting methods are based on binary classification. By the likelihood metric (Table 3), the Bayesian network method performs best on four datasets, comparably to MLN_Boost on Mutagenesis, and slightly worse than both boosting methods on the two largest datasets. By the precision-recall metric (Table 4), the Bayesian network method performs substantially better on four datasets, identically on Hepatitis, and substantially worse on Mutagenesis.

Combining these results, for most of our datasets the Bayesian network method has comparable accuracy and much faster learning. This is satisfactory because boosting is a powerful method that achieves accurate predictions by producing a tailored local model for each target predicate. By contrast, Bayesian network learning simultaneously constructs a joint model for all predicates, and uses simple maximum likelihood estimation for parameter values. We conclude that *Bayesian network learning scales much better to large datasets, and provides competitive accuracy in predictions.*

8.3 Comparison of Model Structures

Boosting is known to lead to very accurate classification models in general [3]. For propositional data, a Bayesian network classifier with maximum likelihood estimation for parameter values is a reasonable baseline method [14], but we would expect less accuracy than from a boosted ensemble of regression trees. Therefore the predictive performance of our RDN models is not due to the log-linear equation (1), but due to the more powerful features that Bayesian network learning finds in relational datasets. These features involve longer chains of relationships than we observe in the boosting models.[7] The ability to find complex patterns involving longer relationship chains comes from the lattice search strategy, which in turn depends on the scalability of model evaluation in order to explore a complex space of relationship chains. Table 5 reports results that quantitatively confirm this analysis.

For each database, we selected the target PRV where RDN-Bayes shows the greatest predictive advantage over RDN-Boost (shown as Δ CLL and Δ AUC-PR). We then compute how many more PRVs the RDN-Bayes model uses to predict the target predicate than the RDN-Boost model, shown as Δ Predicates. This number can be as high as 11 more PRVs (for Mondial). We also compare how many more population variables are contained in the Markov blanket of the RDN-Bayes model, shown as Δ Variables. In terms of database tables, the number of population variables measures how many related tables are used for prediction in addition to the target table. This number can be as high as 2 (for IMDb and Hepatitis). To illustrate Figure 3 shows the parents (Markov blanket) of target node $gender(\mathbb{U})$ from IMDb in the RDN-Boost and RDN-Bayes models. The RDN-Bayes model introduces 4 more parents and 2 more variables, *Movie* and *Actor*. These two variables correspond to a relationship chain of length 2. Thus BN learning discovers that the gender of a user can be predicted by the gender of actors that appear in movies that the user has rated.

Table 5. Difference in Markov blankets between RDN_Bayes and RDN_Boost. Δx = (x for RDN_Bayes - x for RDN_Boost). RDN_Bayes predicts a target more successfully because it uses more predicates and those predicates contain more first-order variables.

Database	Target	Δ Predicates	Δ Vars.	Δ CLL	Δ AUC-PR
Mondial	religion	11	1	0.58	0.30
IMDb	gender	4	2	0.30	0.68
UW-CSE	student	4	1	0.50	0.55
Hepatitis	sex	4	2	0.20	0.25
Mutagenesis	ind1	5	1	0.56	0.22
MovieLens	gender	1	1	0.26	0.26

[7] Kok and Domingos emphasize the importance of learning clauses with long relationship chains [27].

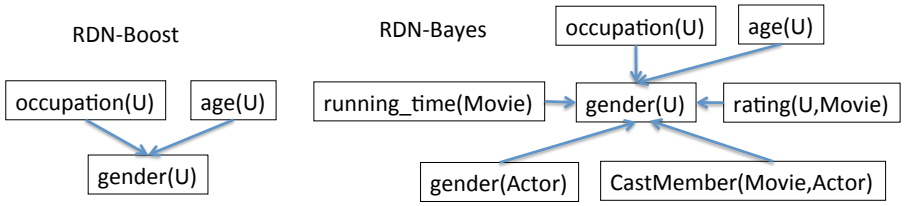

Fig. 3. The parents of target *gender*(𝕌) in the models discovered by RDN_Boost (left) and RDN_Bayes (right).

9 Combining Bayesian Network Learning and Gradient Boosting

Gradient Boosting is a very different approach from Bayesian network conversion, in several respects: (1) Different model type: single-model Bayesian network vs. ensemble of regression trees. (2) Different language bias: the learn-and-join structure learning algorithm considers nodes with population variables only, whereas a novel aspect of structure learning with the boosting methods is that they allow both variables and constants. (3) Different learning methods: local heuristic search to optimize a model selection score, vs. boosting.

Given these fundamental differences, it is not surprising that our experiments show different strengths and limitations for each approach. Strengths of Bayes nets include: (1) Speed through fast model evaluation, which facilities exploring complex cross-table correlations that involve long chains of relationships. (2) Interpretability of the conditional probability parameters. (3) Learning easily extends to attributes with more than two possible values. Strengths of boosting include: (1) Potential greater accuracy through the use of an ensemble of regression trees. (2) Exploring a larger space of statistical-relational patterns that include both first-order variables and constants. These two approaches can be combined in several natural ways to benefit from their mutual strengths.

1. Fast Bayesian network learning methods can be used to select features. Regression tree learning should work much faster when restricted to the BN Markov blanket of a target node. The mode declaration of the Boostr system support adding background knowledge about predictive predicates. This would facilitate exploring longer relationship chains within the boosting framework.
2. The Bayesian network can provide an initial dependency network for the boosting procedure. Gradient boosting can be used in place of maximum likelihood estimation to improve the conditional probability models. Boosting can be applied to improve the estimate of the Bayesian network parameters (node conditional distribution given parents) or of the dependency network parameters (node conditional distribution given parents). It is well-known

that decision trees can improve the estimation of Bayesian network parameters [10]; a tree ensemble should provide an even more accurate model. Using boosting for local probability models would leverage its ability to learn statistical patterns with constants rather than first-order variables only.

3. Functional gradient boosting can be used with proportions as feature functions rather than counts, to avoid ill-conditioned learning with feature functions of different magnitudes.

10 Related Work

Dependency networks were introduced by Heckerman et al. [15] and extended to relational data by Neville and Jensen [34]. Heckerman et al. compare Bayesian, Markov and dependency networks for nonrelational data [15]. Neville and Jensen compare Bayesian, Markov and dependency networks for relational data, including the scalability advantages of Bayesian network learning [34, Sec.8.5.1].

Bayesian networks. There are several proposals for defining directed relational template models, based on graphs with directed edges or rules in clausal format [19, 11]. Defining the probability of a child node conditional on multiple instantiations of a parent set requires the addition of combining rules [19] or aggregation functions [11]. Combining rules such as the arithmetic mean [33] combine global parameters with a local scaling factor, as does our log-linear model. In terms of combining rules, our model uses the *geometric mean* rather than the arithmetic mean.[8] To our knowledge, the geometric mean has not been used before as a combining rule for relational data.

Markov Networks. Markov Logic Networks (MLNs) provide a logical template language for undirected graphical models. Richardson and Domingos propose transforming a Bayesian network to a Markov Logic network using moralization, with log-conditional probabilities as weights [6]. This is also the standard BN-to-MLN transformation recommended by the Alchemy system [1]. A discriminative model can be derived from any MLN [6]. The structure transformation was used in previous work [39], where MLN parameters were learned, not computed in closed-form from BN parameters. The local probability distributions derived from an MLN obtained from converting a Bayesian network are the same as those defined by our log-linear Formula 1, *if* counts replace proportions as feature functions [38]. Since the local probability distributions derived from an MLN are consistent, our main Theorem 1 entails that in general, there is no MLN whose log-linear local models are equivalent to our log-linear local models with proportions as feature functions.[9]

[8] The geometric mean of a list of numbers $x_1, \ldots, x_n$ is $(\prod_i x_i)^{1/n}$. Thus geometric mean = exp(average (logs)).

[9] A preliminary version of this paper was presented at the StarAI 2012 workshop. A second version of this paper was presented at ILP 2014 but not published in the conference proceedings.

11 Conclusion and Future Work

Relational dependency networks offer important advantages for modelling relational data. They can be learned quickly by first learning a Bayesian network, then performing a closed-form transformation of the Bayesian network to a dependency network. The key question is how to transform BN parameters to DN parameters. We introduced a relational generalization of the standard propositional BN log-linear equation for the probability of a target node conditional on an assignment of values to its Markov blanket. The new log-linear equation uses a sum of expected values of BN log-conditional probabilities, with respect to a random instantiation of first-order variables. This is equivalent to using feature instantiation proportions as feature functions. Our main theorem provided a necessary and sufficient condition for when the local log-linear equations for different nodes are mutually consistent. On six benchmark datasets, learning RDNs via BNs scaled much better to large datasets than state-of-the-art functional gradient boosting methods, and provided competitive accuracy in predictions.

Future Work. The boosting approach to constructing a dependency network by learning a collection of discriminative models is very different from learning a Bayesian network. There are various options for hybrid approaches that combine the strengths of both. (1) Fast Bayesian network learning can be used to select features. Discriminative learning methods should work faster restricted to the BN Markov blanket of a target node. (2) The Bayesian network can provide an initial dependency network structure. Gradient boosting can then be used to fine-tune local distribution models.

Appendix: Proof of Consistency Characterization

This appendix presents a proof of Theorem 1. The theorem says that a dependency network derived from a template Bayesian network is consistent if and only if the Bayesian network is edge-consistent. We begin by showing that Bayesian network edge-consistency is sufficient for dependency network consistency. This is the easy direction. That edge-consistency is also necessary requires several intermediate results.

11.1 Edge-Consistency is Sufficient for Consistency

Edge consistency entails that each grounding of a node determines a unique grounding of both its parents and its children in the Bayesian network. Thus the ground dependency network is composed of disjoint dependency networks, one for each grounding. Each of the ground disjoint dependency networks is consistent, so a joint distribution over all can be defined as the product of the joint probabilities of each ground dependency network. The formal statement and proof is as follows.

Proposition 1. *If a template Bayesian network is edge-consistent, then the derived dependency network is consistent.*

Proof. Heckermann *et al.* [15] showed that a dependency network is consistent if and only if there is a Markov network with the same graphical structure that agrees with the local conditional distributions. We argue that given edge-consistency, there is such a Markov network for the derived dependency network. This Markov network is obtained by moralizing and then grounding the Bayesian network [6]. Given edge-consistency, for each ground target node, each family of the ground target node has a unique grounding. Thus the relevant family counts are all either 1 or 0 (0 if the family configuration is irrelevant). The Markov network is now defined as follows: Each grounding of a family in the template Bayesian network is a clique. For an assignment of values $U^* = u, \mathrm{Pa}(U)^* = \boldsymbol{u}_{pa}$ to a ground family, the clique potential is 1 if the assignment is irrelevant, and $\theta(U = u|\mathrm{Pa}(U) = \boldsymbol{u}_{pa})$ otherwise. It is easy to see that the conditional distributions induced by this Markov network agree with those defined by Equation 1, given edge-consistency.

11.2 Edge-Consistency is Necessary for Consistency

This direction requires a mild condition on the structure of the Bayesian network: it must not contain a redundant edge [35]. An edge $T_1 \rightarrow T_2$ is redundant if for every value of the parents of T_2 excluding T_1, every value of T_1 is conditionally independent of every value of T_2. Less formally, given the other parents, the node T_1 adds no probabilistic information about the child node T_2. Throughout the remainder of the proof, we assume that the template Bayesian network contains no redundant edges. Our proof is based on establishing the following theorem.

Theorem 2. *Assume that a template BN contains at least one edge e_1 such that the parent and child do not contain the same set of population variables. Then there exists an edge e_2 (which may be the same as or distinct from e_1) from parent T_1 to child T_2, ground nodes T_1^* and T_2^*, and a query conjunction Λ^* such that: the ground nodes T_1^* and T_2^* have mutually inconsistent conditional distributions $\theta(T_1^*|\Lambda^*)$ and $\theta(T_2^*|\Lambda^*)$ as defined by Equation 1.*

The query conjunction Λ^* here denotes a complete specification of all values for all ground nodes except for T_1^* and T_2^*. Theorem 2 entails the necessity direction of Theorem 1 by the following argument. Suppose that there is a joint distribution p that agrees with the conditional distributions of the derived dependency network. Then for every query conjunction Λ^*, and for every assignment of values t_1 resp. t_2 to the ground nodes, we have that $p(T_1^* = t_1|T_2^* = t_2, \Lambda^*)$ and $p(T_2^* = t_2|T_1^* = t_1, \Lambda^*)$ agree with the log-linear equation 1. Therefore, the conditional distributions $p(T_1^*|T_2^*, \Lambda^*)$ and $p(T_2^*|T_1^*, \Lambda^*)$ must be mutually consistent. Theorem 2 asserts that for every (non-redundant) edge-inconsistent template BN, we can find a query conjunction and two ground nodes such that the conditional distributions of the ground nodes given the query conjunction are not mutually consistent. Therefore there is no joint distribution that is consistent with all the conditional distributions defined by the log-linear equations, which establishes the necessity direction of the main theorem 1.

Properties of the template BN and the input query Λ^*. We begin by establishing some properties of the template BN and the query conjunction that are needed in the remainder of the proof.

The inconsistency of the BN networks arises when a parent and a child ground node have different relevant family counts. The next lemma shows that this is possible exactly when the template BN is properly relational, meaning it relates parents and children from different populations.

Lemma 1. *The following conditions are equivalent for a template edge $T_1 \rightarrow T_2$.*

1. *The parent and child do not contain the same population variables.*
2. *It is possible to find a grounding γ for both parent and child, and an assignment Λ^* to all other nodes, such that the relevant family count for the T_2 family differs for $T_1^* = \gamma T_1$ and $T_2^* = \gamma T_2$.*

Proof. If the parent and child contain the same population variables, then there is a 1-1 correspondence between groundings of the child and groundings of the parents. Hence the count of relevant family groundings is the same for each, no matter how parents and child are instantiated. If the parent and child do not contain the same population variables, suppose without loss of generality that the child contains a population variable $\mathbb{A}$ not contained in the parent. Choose a common grounding γ for the parents and child node. For the ground child node, γT_2, let γ be the only family grounding that is relevant, so the relevant count is 1. For the ground parent node, there is at least one other grounding of the child node T_2' different from γT_2 since T_2 contains another population variables. Thus it is possible to add another relevant family grounding for γT_1, which means that the relevant count is at least 2.

The proof proceeds most simply if we focus on template edges that relate different populations and no common children.

Definition 2. *An template edge $T_1 \rightarrow T_2$ is **suitable** if*

1. *The parent and child do not contain the same population variables.*
2. *The parent and child have no common edge.*

The next lemma shows that focusing on suitable edges incurs no loss of generality.

Lemma 2. *Suppose that a template BN contains an edge such that the parent and child do not contain the same population variables. Then the template BN contains a suitable edge.*

Proof. Suppose that there is an edge satisfying the population variable condition. Suppose that the parent and child share a common child. Since the edge satisfies the condition, the set of population variables in the common child differs from at least one of T_1, T_2. Therefore there is another edge from one of $T_1 \rightarrow T_2$ as parent to a new child that satisfies the population variable condition. If this edge is not suitable, there must be another shared child. Repeating this argument, we eventually arrive at an edge satisfying the population variable condition where the child node is a sink node without children. This edge is suitable.

Consider a suitable template edge $T_1 \rightarrow T_2$ that produces a bidirected ground edge $T_1^* \leftrightarrow T_2^*$. For simplicity we assume that T_1 and T_2 are binary variables with domain $\{\mathrm{T}, \mathrm{F}\}$. (This incurs no loss of generality as we can choose a database $\mathcal{A}^*$ in which only two values occur.) Let $\mathrm{Pa}(T_2)$ be the parents of T_2 other than T_1. Since the template edge is not redundant [35], there is a parent value setting $\mathrm{Pa}(T_2) = \mathbf{pa}$ such that T_1 and T_2 are conditionally dependent given $\mathrm{Pa}(T_2) = \mathbf{pa}$. This implies that the conditional distribution of T_1 is different for each of the two possible values of T_2:

$$\frac{\theta(T_2 = \mathrm{F}|T_1 = \mathrm{F}, \mathbf{pa})}{\theta(T_2 = \mathrm{T}|T_1 = \mathrm{F}, \mathbf{pa})} \neq \frac{\theta(T_2 = \mathrm{F}|T_1 = \mathrm{T}, \mathbf{pa})}{\theta(T_2 = \mathrm{T}|T_1 = \mathrm{T}, \mathbf{pa})}. \tag{1}$$

Let $\mathcal{A}^*$ denote an assignment of values to all ground nodes other than the target nodes T_1^* and T_2^*. We assume that the input query $\mathcal{A}^*$ assigns different relevant family counts N_1 to T_1^* and N_2 to T_2^*. This is possible according to Lemma 1.

Lowd's Equation and Relevant Family Counts The log-linear equation 1, specifies the conditional distribution of each target node given $\mathcal{A}^*$ and a value for the other target node. We keep the assignment $\mathcal{A}^*$ fixed throughout, so for more compact notation, we abbreviate the conditional distributions as

$$p({T_1}^* = t_1|{T_2}^* = t_2) \equiv P({T_1}^* = t_1|{T_2}^* = t_2, \mathcal{A}^*)$$

and similarly for $P({T_1}^* = t_1|{T_2}^* = t_2, \mathcal{A}^*)$.

On the assumption that the dependency network is consistent, there is a joint distribution over the target nodes conditional on the assignment that agrees with the conditional distribution:

$$\frac{p({T_1}^* = t_1, {T_2}^* = t_2)}{p({T_2}^* = t_2)} = p({T_1}^* = t_1|T_2)^*$$

and also with the conditional $p({T_2}^* = t_2|{T_1}^* = t_1)$.

Lowd [29] pointed out that this joint distribution satisfies the equations

$$\frac{p(\mathrm{F},\mathrm{F})}{p(\mathrm{T},\mathrm{F})} \cdot \frac{p(\mathrm{T},\mathrm{F})}{p(\mathrm{T},\mathrm{T})} = \frac{p(\mathrm{F},\mathrm{F})}{p(\mathrm{T},\mathrm{T})} = \frac{p(\mathrm{F},\mathrm{F})}{p(\mathrm{F},\mathrm{T})} \cdot \frac{p(\mathrm{F},\mathrm{T})}{p(\mathrm{T},\mathrm{T})} \tag{2}$$

Since the ratio of joint probabilities is the same as the ratio of conditional probabilities for the same conditioning event, consistency entails the following constraint on conditional probabilities via Equation (2):

$$\frac{p({T_2}^* = \mathrm{F}|{T_1}^* = \mathrm{F})}{p({T_2}^* = \mathrm{T}|{T_1}^* = \mathrm{F})} \cdot \frac{p({T_1}^* = \mathrm{F}|T_2^* = \mathrm{T})}{p({T_1}^* = \mathrm{T}|{T_2}^* = \mathrm{T})} = \frac{p({T_1}^* = \mathrm{F}|{T_2}^* = \mathrm{F})}{p({T_1}^* = \mathrm{T}|{T_2}^* = \mathrm{F})} \cdot \frac{p({T_2}^* = \mathrm{F}|{T_1}^* = \mathrm{T})}{p({T_2}^* = \mathrm{T}|{T_1}^* = \mathrm{T})} \tag{3}$$

We refer to Equation 3 as *Lowd's equation*. The idea of our proof is to show that Lowd's equations are satisfied only if the relevant family counts for the target nodes are the same. According to the log-linear equation, each conditional

probability is proportional to a product of BN parameters. The first step is to show that in Lowd's equation, all BN parameter terms cancel out except for those that are derived from the family that comprises T_1^* and their T_2^* and their common grounding.

Lemma 3. *The conditional probabilities for the target nodes can be written as follows:*

$$P({T_2}^* = t_2 | T_1^* = t_1, \Lambda^*) \propto \theta(T_2 = t_2 | T_1 = t_1, \mathbf{pa})^{(N/N_2 + M_{T_2=t_2}/N_2)} \cdot \pi_{T_2=t_2} \quad (4)$$

where $M_{T_2=t_2}$ *and* $\pi_{T_2=t_2}$ *depend only on* t_2 *and not on* t_1 *and*

$$P({T_1}^* = t_1 | T_2^* = t_2, \Lambda^*) \propto \theta(T_2 = t_2 | T_1 = t_1, \mathbf{pa})^{(N/N_1 + M_{T_1=t_1}/N_1)} \cdot \pi_{T_1=t_1} \quad (5)$$

where $M_{T_1=t_1}$ *and* $\pi_{T_1=t_1}$ *depend only on* t_1 *and not on* t_2.

Proof Outline. This is based on analysing the different types of families that appear in the log-linear equation and their groundings. We omit this straightforward analysis to simplify the proof; the details are available from [41].

Lemma 4. *Suppose that conditions* (4) *and* (5) *of Lemma 3 hold. Then Lowd's Equation* (3) *holds if and only if* $N_1 = N_2$.

Proof. Observe that in Equation (3), each term on the left has a corresponding term with the same value for the target node assignment and the opposing conditioning assignment. For instance, the term $p({T_2}^* = \mathrm{F} | {T_1}^* = \mathrm{F})$ on the left is matched with the term $p({T_2}^* = \mathrm{F} | {T_1}^* = \mathrm{T})$ on the right. This means that the products in the log-linear expression are the same on both sides of the equation except for those factors that depend on *both* t_1 and t_2. Continuing the example, the factors

$$\theta(T_2 = \mathrm{F} | T_1 = \mathrm{F}, \mathbf{pa})^{(M_\mathrm{F}/N_2)} \cdot \pi_{T_2=t_2}$$

on the left equal the factors

$$\theta(T_2 = \mathrm{F} | T_1 = \mathrm{T}, \mathbf{pa})^{(M_{T_1=t_1}/N_2)} \cdot \pi_{T_2=t_2}$$

on the right side of the equation. They therefore cancel out, leaving only the term

$$\theta(T_2 = \mathrm{F} | T_1 = \mathrm{F}, \mathbf{pa})^{N/N_2}$$

on the left and the term

$$\theta(T_2 = \mathrm{F} | T_1 = \mathrm{F}, \mathbf{pa})^{N/N_2}$$

on the right. Lowd's equation can therefore be reduced to an equivalent constraint with only such BN parameter terms. For further compactness we abbreviate such terms as follows

$$\theta(t_2 | t_1) \equiv \theta(T_2 = t_2 | T_1 = t_1, \mathbf{pa}).$$

With this abbreviation, the conditions of Lemma 3 entail that Lowd's equation 3 reduces to the equivalent expressions.

$$\frac{\theta(\mathrm{F}|\mathrm{F})^{N/N_2}}{\theta(\mathrm{T}|\mathrm{F})^{N/N_2}} \cdot \frac{\theta(\mathrm{T}|\mathrm{F})^{N/N_1}}{\theta(\mathrm{T}|\mathrm{T})^{N/N_1}} = \frac{\theta(\mathrm{F}|\mathrm{F})^{N/N_1}}{\theta(\mathrm{F}|\mathrm{T})^{N/N_1}} \cdot \frac{\theta(\mathrm{F}|\mathrm{T})^{N/N_2}}{\theta(\mathrm{T}|\mathrm{T})^{N/N_2}} \tag{6}$$

$$(\frac{\theta(\mathrm{F}|\mathrm{F})}{\theta(\mathrm{T}|\mathrm{F})})^{(N/N_2 - N/N_1)} = (\frac{\theta(\mathrm{F}|\mathrm{T})}{\theta(\mathrm{T}|\mathrm{T})})^{(N/N_2 - N/N_1)} \tag{7}$$

By the nonredundancy assumption (1) on the BN parameters, we have

$$\frac{\theta(\mathrm{F}|\mathrm{F})}{\theta(\mathrm{T}|\mathrm{F})} \neq \frac{\theta(\mathrm{F}|\mathrm{T})}{\theta(\mathrm{T}|\mathrm{T})}$$

so Equation 7 implies that

$$N_1 = N_2,$$

which establishes the lemma.

Theorem 2 now follows as follows: Lemma 1 entails that if the dependency network is consistent, the log-linear equations satisfy Lowd's equation with the bidirected ground edge $T_1^* \leftrightarrow T_2^*$ and the query conjunction Λ^* that satisfies the BN non-redundancy condition. Lemmas 3 and 2 show that if the template BN is relational, it must contain a suitable edge $T_1 \rightarrow T_2$. Lemma 4 together with Lowd's equation entails that the relevant counts for T_1^* and T_2^* must then be the same. But the query conjunction Λ^* was chosen so that the relevant counts are different. This contradiction shows that Lowd's equation is unsatisfiable, and therefore no joint distribution exists that is consistent with the BN conditional distributions specified by the log-linear Equation 1. Since Theorem 2 entails Theorem 1, our proof is complete.

References

1. Alchemy Group. Frequently asked questions. URL = `http://alchemy.cs.washington.edu/`.
2. Yoshua Bengio, Eric Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. In *ICML 2014*, pages 226–234, 2014.
3. Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
4. D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is np-hard. *JMLR*, 5:1287–1330, 2004.
5. David Maxwell Chickering and Christopher Meek. Finding optimal Bayesian networks. In *UAI*, pages 94–102, 2002.
6. Pedro Domingos and Daniel Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan and Claypool Publishers, 2009.
7. Pedro Domingos and Matthew Richardson. Markov logic: A unifying framework for statistical relational learning. In *Introduction to Statistical Relational Learning* [12].

8. Jörg Flum and Martin Grohe. *Parameterized complexity theory*, volume 3. Springer, 2006.
9. Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI*, pages 1300–1309. Springer-Verlag, 1999.
10. Nir Friedman and Moises Goldszmidt. Learning Bayesian networks with local structure. In *NATO ASI on Learning in graphical models*, pages 421–459, 1998.
11. Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Benjamin Taskar. Probabilistic relational models. In *Introduction to Statistical Relational Learning* [12], chapter 5, pages 129–173.
12. Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning*. MIT Press, 2007.
13. Lise Getoor Getoor, Nir Friedman, and Benjamin Taskar. Learning probabilistic models of relational structure. In *ICML*, pages 170–177. Morgan Kaufmann, 2001.
14. Daniel Grossman and Pedro Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *ICML*, page 46, New York, NY, USA, 2004. ACM.
15. David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, Carl Kadie, and Pack Kaelbling. Dependency networks for inference, collaborative filtering, and data visualization. *JMLR*, 1:49–75, 2000.
16. Reimar Hofmann and Volker Tresp. Nonlinear markov networks for continuous variables. pages 521–527. MORGAN KAUFMANN PUBLISHERS, 1998.
17. Geoff Hulten, David Maxwell Chickering, and David Heckerman. Learning bayesian networks from dependency networks: A preliminary study. In *in Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, Key West, FL*, 2003.
18. Seyed Mehran Kazemi, David Buchman, Kristian Kersting, Sriraam Natarajan, and David Poole. Relational logistic regression. In *Principles of Knowledge Representation and Reasoning KR*, 2014.
19. Kristian Kersting and Luc de Raedt. Bayesian logic programming: Theory and tool. In *Introduction to Statistical Relational Learning* [12], chapter 10, pages 291–318.
20. Hassan Khosravi, Tong Man, Jianfeng Hu, Elwin Gao, and Oliver Schulte. Learn and join algorithm code. URL = `http://www.cs.sfu.ca/~oschulte/jbn/`.
21. Hassan Khosravi, Oliver Schulte, Tong Man, Xiaoyuan Xu, and Bahareh Bina. Structure learning for Markov logic networks with many descriptive attributes. In *AAAI*, pages 487–493, 2010.
22. Tushar Khot, Sriraam Natarajan, Kristian Kersting, and Jude W. Shavlik. Learning Markov logic networks via functional gradient boosting. In *ICDM*, pages 320–329, 2011.
23. Tushar Khot, Jude Shavlik, and Sriraam Natarajan. Boostr, 2013. URL = `http://pages.cs.wisc.edu/~tushar/Boostr/`.
24. Angelika Kimmig, Lilyana Mihalkova, and Lise Getoor. Lifted graphical models: a survey. *Machine Learning*, pages 1–45, 2014.
25. Arno J Knobbe. *Multi-relational data mining*, volume 145. Ios Press, 2006.
26. Stanley Kok and Pedro Domingos. Learning the structure of Markov logic networks. In Luc De Raedt and Stefan Wrobel, editors, *ICML*, pages 441–448. ACM, 2005.
27. Stanley Kok and Pedro Domingos. Learning Markov logic networks using structural motifs. In *ICML*, pages 551–558, 2010.
28. Wim Van Laer and Luc de Raedt. How to upgrade propositional learners to first-order logic: A case study. In *Relational Data Mining*. Springer Verlag, 2001.

29. Daniel Lowd. Closed-form learning of Markov networks from dependency networks. In *UAI*, 2012.
30. Daniel Lowd and Pedro Domingos. Efficient weight learning for Markov logic networks. In *PKDD*, pages 200–211, 2007.
31. Andrew W. Moore and Mary S. Lee. Cached sufficient statistics for efficient machine learning with large datasets. *J. Artif. Intell. Res. (JAIR)*, 8:67–91, 1998.
32. Sriraam Natarajan, Tushar Khot, Kristian Kersting, Bernd Gutmann, and Jude W. Shavlik. Gradient-based boosting for statistical relational learning: The relational dependency network case. *Machine Learning*, 86(1):25–56, 2012.
33. Sriraam Natarajan, Prasad Tadepalli, Thomas G. Dietterich, and Alan Fern. Learning first-order probabilistic models with combining rules. *Annals of Mathematics and Artifical Intelligence*, 54(1-3):223–256, 2008.
34. Jennifer Neville and David Jensen. Relational dependency networks. *JMLR*, 8:653–692, 2007.
35. Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
36. Veronika Peralta. Extraction and integration of MovieLens and IMDB data. Technical report, Laboratoire PRiSM, Universite de Versailles, 2007.
37. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2010.
38. Oliver Schulte. A tractable pseudo-likelihood function for Bayes nets applied to relational data. In *SIAM SDM*, pages 462–473, 2011.
39. Oliver Schulte and Hassan Khosravi. Learning graphical models for relational data via lattice search. *Machine Learning*, 88(3):331–368, 2012.
40. Oliver Schulte, Hassan Khosravi, Arthur Kirkpatrick, Tianxiang Gao, and Yuke Zhu. Modelling relational statistics with bayes nets. *Machine Learning*, 94:105–125, 2014.
41. Oliver Schulte, Zhensong Qian, Arthur E. Kirkpatrick, Xiaoqian Yin, and Yan Sun. Fast learning of relational dependency networks. *CoRR*, abs/1410.7835, 2014.
42. Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning* [12], chapter 4, pages 93–127.
43. Benjamin Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In *UAI*, pages 485–492. Morgan Kaufmann Publishers Inc., 2002.
44. J. D. Ullman. *Principles of Database S¡ystems*. W. H. Freeman & Co., 2 edition, 1982.
45. Moshe Y. Vardi. On the complexity of bounded-variable queries. In *PODS*, pages 266–276. ACM Press, 1995.