# 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-PR | CLL |
| RDN\_Boost | 0.59 | -0.65 |
| MLN\_Boost | 0.56 | -1.48 |
| RDN\_Bayes | **1.0** | **-0.43** |

*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.