|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ILP 2014 (author)**   |  |  |  | | --- | --- | --- | | Submission 22 | ILP 2014 | EasyChair |   ILP 2014 Submission 22  If you want to **change any information** about your paper or withdraw it, use links in the upper right corner.  For all questions related to processing your submission you should contact the conference organizers. [Click here to see information about this conference.](https://www.easychair.org/conferences/conference_info.cgi?track=94422;a=6589896)  All **reviews sent to you** can be found at the bottom of this page.   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Paper 22** | | | | | | | | Title: | Fast Learning of Relational Dependency Networks | | | | | | | Paper: | [PDF](https://www.easychair.org/conferences/submission_download.cgi?submission=1845132;track=94422;a=6589896) (May 15, 00:17 GMT) | | | | | | | Track: | ILP 2014: Long papers | | | | | | | Author keywords: | Relational Dependency Network(RDN)  Bayesian Network(BN)  Multi-Relational Data  BN-to-RDN Closed-Form Transform  Fast Learning | | | | | | | 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 Bayes net learning can provide fast RDN learning.  The BN-to-RDN transform comprises a simple, local adjustment of the Bayes net structure and a closed-form transform of  the Bayes net parameters. This method can learn an RDN for a dataset with a million tuples in minutes.  We empirically compare our approach to state-of-the art RDN learning methods that use functional gradient boosting,  on seven benchmark datasets.  Learning RDNs via BNs scales much better to large datasets than learning RDNs with boosting, and provides competitive accuracy in predictions. | | | | | | | Time: | May 11, 00:37 GMT | | | | | | | **Authors** | | | | | | | | first name | last name | Email | country | organization | Web site | corresponding? | | Oliver | Schulte | oschulte@cs.sfu.ca | Canada | Simon Fraser University | <http://www.cs.sfu.ca/~oschulte/> | ✔ | | Zhensong | Qian | [zqian@sfu.ca](mailto:zqian@sfu.ca) | Canada | simon fraser university | <http://www.sfu.ca/~zqian> | ✔ | | Arthur | E. Kirkpatrick | [ted@sfu.ca](mailto:ted@sfu.ca) | Canada | simon fraser university |  |  | | Xiaoqian | Yin | xiaoqian\_yin@sfu.ca | Canada | Simon Fraser University |  |  | | Yan | Sun | [sunyans@sfu.ca](mailto:sunyans@sfu.ca) | Canada | Simon Fraser University |  |  |   Reviews   |  |  | | --- | --- | | **Review 3** | | | Overall evaluation: | **3**: (weak accept in LNCS) | | Reviewer's confidence: | **3**: (medium) | | Review: | The paper presents an approach for computing RDN from large dataset by  using a two-step process of generating first a Bayesian network structure with parameters  and then transforming this structure into a RDN.  **The novel contribution is the transformation of BN parameters into DN parameters**.  The proposed method uses the notion of Markov blanket.   The approach has been applied to benchmarking large databases of difference sizes including the MovieLens dataset  containing 1M records.  It is shown that the predictive accuracy compares well with that of existing state-of-the-art function gradient boosting methods  but it is can scale up to larger datasets.   The paper clearly provides a valuable contribution to the area of Machine Learning,  although in the experimental evaluation section it is not clear what it is learned.  The paper is generally **well written** although it lacks of explanations of the notations used.   - For example, in Figure 1 it Is not clear why some functors are have capital initial and others don’t.  - It is not clear what the dashed arrows mean in the transformation of BN to Dn. The authors should explain this further.  - In Section 4 Definition 1 provides the main technical contribution of this work: i.e. how to generate the DN parameters  from given BN parameters. It is therefore important that provide detailed explanation so to allow readers with limited  technical background in this area to have a good understanding of the benefits of this method for computing parameters.  An example should be given that exemplifies the notion here defined, and instantiate all the notations used.  Minor typos:  1) page 6, last line: “new new” 2) page 7: include the www link in the references. 3) page 11: “learning a Bayesian network learning” |  |  |  | | --- | --- | | **Review 2** | | | Overall evaluation: | **4**: (extended version in LNCS) | | Reviewer's confidence: | **4**: (high) | | Review: | The paper illustrates how to make use of relational Bayesian network learning for learning relational dependency networks.   A rather technical question, in the original work on dependency networks  Heckerman et al. actually spend quite some space on consistency, say that not every DN is consistent.  Now, given you start with learning a BN, does that mean that the induced DNs are always consistent? This may also explain the different performances across the different benchmarks.   Furthermore, the text uses relational models and propositional models in an interchangeable way. I think it would be better to carefully say exactly which type of model is meant in each situation. Why? Well, while I agree that a BN can easily turned into a DN, I am not so sure about relational BNs. This very much depends on the semantics.  For instance, if you allow for existentially quantified variables or equality constraints, then we cannot just link nodes.  We have to ground it. And actually Section 4 is all about this.   Finally, and maybe a potential downside is the missing discussion of why the proposed learning method is performing so well.  Is it the bias of the used relational BN learner? Is it the specific formula explained in Section 4?  It appears to me that this implements a form of regularization that may explain the better predictive performance on some benchmarks.  Or, is it because of a more efficient implementation? I mean how much is the slower running time of the boosting approaches due to its implementation? How many iterations does it take actually? Could we stop early?  Anyhow, these are never-ending questions and I like the empirical observation that transformations of different relational models can actually “boost” performance. This is an interesting take-away story.  Still, I would like to encourage the authors to spend more space on a discussion trying to explain the difference.  In the end, you have only one large dataset and here I can imagine that Hoeffding trees could actually speed up the boosting approaches a lot.  http://ilp11.doc.ic.ac.uk/short\_papers/ilp2011\_submission\_33\_old.pdf This may in a boosting context even have a regularization effect as we do not „memorize all the training cases“ as explained in the  above extended abstract. I think the authors should really touch upon an explanation.  To summarize, an interesting idea that shows some benefit.  Unfortunately, only one larger dataset has been used so that we do not understand the scaling behavior well. And the "why" is not discussed. |  |  |  | | --- | --- | | **Review 1** | | | Overall evaluation: | **4**: (extended version in LNCS) | | Reviewer's confidence: | **3**: (medium) | | Review: | This paper presents an approach to compute a relational dependency network by first learning a Bayesian network  and then transforming it into a relational dependency network.  Experiments investigate both accuracy and scalability of the proposed approach compared to benchmark competitors.  Some suggestions to improve this manuscript are listed below: 1) A theoretical study of the time complexity of the presented approach should be reported in Section 3. 2) The pseudo code of the several steps shown in the program flow of Figure 2 should be included in the paper.  Some examples can be used to describe these steps. 3) A statistical analysis (e.g. the pairwise Wilcoxon test) of the results reported in Tables 3-4 should be added to the paper,  in order to determine the statistical significance of the reported comparisons. | |
|  |