PRIMULA-3 for Probabilistic Modeling and Reasoning on Graph Data

Raffaele Pojer (✉), Manfred Jaeger

Aalborg University, Denmark {rafpoj,jaeger}@cs.aau.dk

Abstract. The PRIMULA system is a versatile software tool for modeling and reasoning with probabilistic relational structures based on the symbolic Relational Bayesian Networks (RBN) language. The new version 3 of PRIMULA extends previous versions by adding support for categorical variables, and by integrating Graph Neural Networks (GNN) as model components into a full generative RBN model, thus combining the predictive power and scalable learning tools of GNNs with the high expressivity and flexible inference capabilities of RBNs.

1 Introduction

Probabilistic modeling of relational data is a cornerstone of modern artificial intelligence, enabling systems to reason under uncertainty in complex, structured domains, such as social or sensor networks, or biological and environmental systems. Relational Bayesian Networks (RBNs) [1] are a powerful framework for working with probabilistic relational models. The PRIMULA software implements the RBN language and provides support for a wide range of learning and reasoning tasks, including: predictive inference comprising standard supervised tasks such as node, link and graph prediction; general probabilistic reasoning for diagnostic inference and decision support, and unsupervised learning, e.g. for community detection. PRIMULA-3 is a revised and updated version that adds as new functionalities: support for categorical variables, and integration with graph neural networks (GNNs) implemented in PyTorch. This integration connects the predictive capabilities of GNNs with the expressive relational logic of RBNs, enabling neuro-symbolic reasoning on graph-structured data. PRIMULA-3 is a *proof-of-concept* software, providing a uniform framework supporting a wide spectrum of graph analysis tasks. It is intended for research and educational purposes by enabling design and experimentation within a rich and versatile framework for modeling and learning with graph data.

2 The PRIMULA-3 System

PRIMULA takes as input a probabilistic model specification in the RBN language (possibly containing GNN components), and a relational domain specification. Together, these inputs define a generative model for probabilistic attributes and relations. Several forms of inference for the resulting models are provided: *exact*

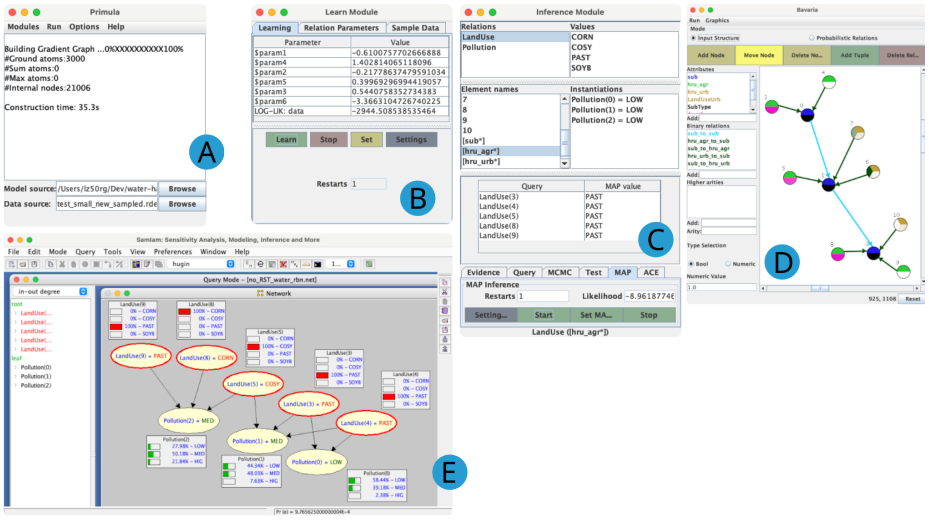

Fig. 1. PRIMULA modules: A: main console, B: learn module for parameter learning, C: inference module for conditional probability and maximum a-posteriori inference, D: module for editing and visualizing relational input domains, E: the external SamIam tool for probabilistic inference by Bayesian networks.

inference by compiling the model into a Bayesian network or arithmetic circuit representation, and applying the dedicated, external SamIam[1] and ACE[2] tools; *approximate* inference with Gibbs sampling for probability computations, and combinatorial optimization for maximum-a-posteriori queries. All inference techniques are equally applicable for models with or without GNN components.

Model specification. We illustrate the RBN language and its GNN integration by the example in Listing 1.1 of an environmental domain with *land* and *water* nodes connected by a spatial *adjacent* relation. Figure 1 D. shows a tiny example domain with node colors representing node types and attributes.

An external GNN model has been trained to model the pollution level based on the agricultural land use at adjacent land units. Lines 3.-6. import this GNN into the RBN model by specifying its Python source and logical signature (input attributes, relation used for message passing, output dimension). Lines 8.-10. are a user-defined function for the profit obtained by the agricultural use of a land unit. Lines 12-15 define a logistic regression model for a Boolean variable indicating whether an intervention at (and around) water node w is necessary to mitigate high pollution or low profit in this area. As shown in [4], a large class of GNN models can also directly be encoded in the native RBN language. However, the ability to also connect to external PyTorch models leads to greater flexibility with regard to GNN architectures, and computational benefits arising from the

[1] `http://reasoning.cs.ucla.edu/samiam/`
[2] `http://reasoning.cs.ucla.edu/ace/`

Listing 1.1. Modeling environmental and economic impact of land use

```
1   LandUse([land]l) = SOFTMAX 1,1,1;
2
3   Pollution([water]w) =  COMPUTEWITHTORCH  <path to PyTorch model>
4                          WITHNUMVALUES 3
5                          FORFREEVARS (w)
6                          USINGRELS LandUse(l) WITHEDGE adjacent(l,w);
7
8   @profit([land]l) = WIF LandUse(l)=corn THEN (2.4*Area(l))
9                      ELSE WIF LandUse(l)=pasture THEN (1.0*Area(l))
10                     ELSE WIF LandUse(l)=soy THEN (1.9*Area(l)) ELSE 0.0;
11
12  Intervention([water]w) = COMBINE 1.8*(Pollution(w)=high),
13                                   0.7*(Pollution(w)=medium),
14                                   -1.1*@profit([land]l)
15                           WITH log-reg FORALL l WHERE adjacent(l,w);
```

greater efficiency of dedicated GNN learners for GNN models. A detailed RBN language documentation and a tutorial example similar to Listing 1.1 is provided with the PRIMULA distribution.

Inference. Given an RBN model instantiated with an input domain the following inference tasks can be solved:

1. Conditional probability queries: given a partial observation of some attributes and relations, what are the conditional probabilities of a specified list of queries? In our environmental domain example: given observed land use, what are the probabilities for pollution levels? For smaller sized problems, these queries can be solved exactly using SamIam and ACE. For larger problems approximate inference by Gibbs sampling is used.
2. MAP queries: given a partial observation of some attributes/relations, what is the most probable joint configuration of a list of query attributes/relations? An example is shown in Figure 1 C, where the most probable land use at all land nodes was queried, given an observation of low pollution at all water nodes. MAP queries are solved by a combinatorial optimization process operating on the *likelihood graph* data structure that also plays a key role in learning.

Figure 2 illustrates a probabilistic inference scenario for an information diffusion model (included in the PRIMULA distribution). The underlying RBN model here encodes the standard *independent cascade* model for information diffusion. We apply the model to the famous Zachary's Karate Club network (Figure 2 A), where node 34 has been labeled as the source node for the diffusion process (indicated by a blue marking in the graph viewer). We can now query for all nodes the probability that they have received the information at time 4, conditioned on the information that node 4 had the information at time 2 (Figure 2 B).

Learning. Parameter learning is supported by gradient descent using LBFGS or Adam. The original learning approach of [2] consists of compiling the model

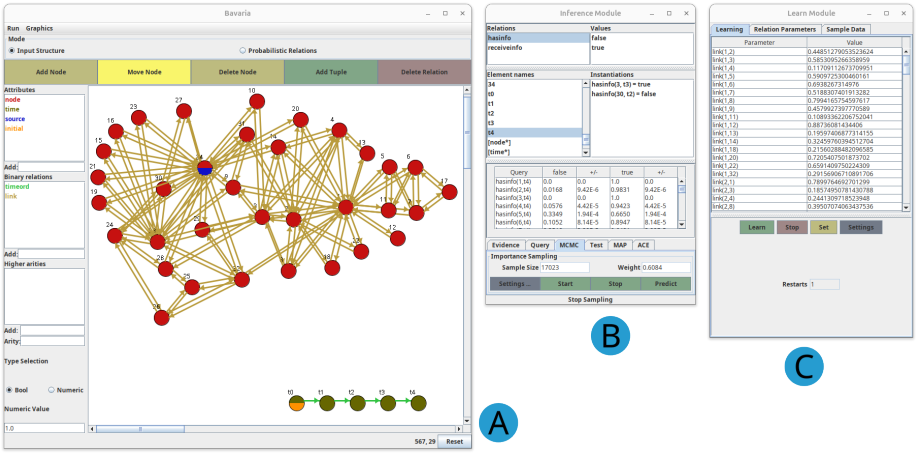

Fig. 2. Information diffusion on the Karate Club network. A: graph viewer/editor, B: inference module, C: learned diffusion probabilities.

into a *likelihood graph* for automatic computation of likelihood values and likelihood gradients. PRIMULA-3 also provides an alternative method that directly computes gradients from the syntactic model representation. The two methods provide different space/time tradeoff characteristics, with the likelihood graph requiring more space, but offering time advantages when the gradient descent requires many iterations, and can benefit from a higher degree of model compilation before learning. With parameter learning, all numerical model parameters can be learned based on a maximum likelihood objective. As a different application of the same underlying optimization routines, one can define numerical node attributes or numerical relations as model parameters. Optimizing the likelihood function for these parameters amounts to learning latent (node/edge) representations. The PRIMULA distribution contains as an application of this approach a multi-factor clustering (or multi-level community detection) of geographic regions in Switzerland based on their plant distribution patterns.

In the information diffusion example of Figure 2, all edges are associated with a probability of information propagation along that edge. Figure 2 shows the result of learning these parameters from a set of sampled diffusion processes.

The PRIMULA *distribution.* PRIMULA-3 is implemented in JAVA and available at `https://github.com/manfred-jaeger-aalborg/primula3`. Both the source code and a platform independent runnable .jar file are provided. The GNN integration is currently only tested on Linux and MacOS platforms. The distribution also contains four tutorial examples demonstrating applications in genetic modeling, information diffusion, community detection, and environmental modeling and decision support. The first three of these examples are pure RBN models, whereas the last includes a GNN component. A video demonstration is available at `https://www.youtube.com/watch?v=6DcWcX-_vA0`.

3 Related tools

Other systems that combine neural networks with logic-symbolic knowledge representation are DeepProbLog [3] [3] and NeurASP[4] [5]. However, these systems differ substantially in terms of the underlying logic-symbolic representations and the way neural components are integrated. The logic element of DeepProbLog and NeurASP is based on logic programming, which, on the one hand, limits expressivity to simple rules, but on the other hand enables via the application of least fixed point or stable model semantics, the modeling of transitive closures of relations, which is outside the scope of PRIMULA-3. The logic of PRIMULA-3 is a generalization of full first-order logic, which allows 'deep' nesting of logic constructs, enabling, among other things, the direct encoding of GNNs in the underlying RBN language. Regarding neural integration, the underlying philosophy in the logic programming-oriented tools is that of a division of labor: the neural components handle low-level tasks related to perception, whereas the symbolic parts handle high-level reasoning. PRIMULA-3 is not based on such an a priori distinction between 'neural' and 'logical' reasoning and their associated relations. Compared to the alternatives, PRIMULA-3 supports a richer class of probabilistic queries, including conditioning on arbitrary observations (including negative facts) and MAP inference.

Acknowledgment. This research has been partially funded by the Villum Investigator Grant S4OS (37819) from Villum Foundation.

References

1. Jaeger, M.: Relational Bayesian networks. In: Proceedings of UAI (1997)
2. Jaeger, M.: Parameter learning for relational Bayesian networks. In: Proceedings of the 24th International Conference on Machine Learning (ICML) (2007)
3. Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., De Raedt, L.: Deepproblog: Neural probabilistic logic programming. Advances in Neural Information Processing Systems **31**, 3749–3759 (2018)
4. Pojer, R., Passerini, A., Jaeger, M.: Generalized reasoning with graph neural networks by relational bayesian network encodings. In: The Second Learning on Graphs Conference. PMLR, vol. 231 (2023)
5. Yang, Z., Ishay, A., Lee, J.: Neurasp: embracing neural networks into answer set programming. In: IJCAI (2021)

[3] `https://github.com/ML-KULeuven/deepproblog`
[4] `https://github.com/azreasoners/NeurASP`