DeepProbLog: Integrating Logic and Learning through Algebraic Model Counting

Robin Manhaeve
KU Leuven
robin.manhaeve@cs.kuleuven.be

Sebastijan Dumančić
KU Leuven
sebastijan.dumancic@cs.kuleuven.be

Angelika Kimmig
Cardiff University
KimmigA@cardiff.ac.uk

Thomas Demeester
Ghent University - imec
thomas.demeester@ugent.be

Luc De Raedt
KU Leuven
luc.deraedt@cs.kuleuven.be

Abstract

The overall goal of neuro-symbolic computation is to integrate high-level reasoning with low-level perception. With DeepProbLog, we show how such an integration can be achieved by using algebraic model counting. Furthermore, we argue that 1) neuro-symbolic computation should integrate neural networks with the two most prominent methods for reasoning, that is, logic and probability, and that 2) neuro-symbolic integrated methods should have the pure neural, logical and probabilistic methods as edge cases. We examine the state-of-the-art with regard to these claims and position our own contribution DeepProbLog in this perspective.[1]

1 Introduction

There is a growing interest in neuro-symbolic computation, that is, in combining high-level reasoning with low-level perception that offers the best of both worlds. The growing consensus that both forms of reasoning are essential to achieve true (artificial) intelligence [Kahneman, 2011] has put the quest for neural symbolic computation [Garcez et al., 2012, 2015, Hammer and Hitzler, 2007] high on the research agenda. While deep learning excels at low-level perception, there is also a growing awareness of its limitations, certainly in terms of reasoning. Despite various attempts to demonstrate reasoning-like behaviour with deep learning [Santoro et al., 2017], artificial neural networks' current reasoning abilities are nowhere close to what is possible with the typical high-level reasoning approaches. The two most prominent frameworks for reasoning are logic and probability. While in the past, they were studied by separate communities in artificial intelligence, a growing body of researchers is working towards their integration, and even aims at combining probability with logic and statistical learning; cf. the areas of statistical relational artificial intelligence [De Raedt et al., 2016, Getoor and Taskar, 2007] and probabilistic logic programming [De Raedt and Kimmig, 2015]. The reasoning abilities of statistical relational artificial intelligence approaches are complementary to the strong pattern-recognition abilities of deep learning. In this paper, we show one approach at achieving this integration through the use of algebraic model counting [Kimmig et al., 2017]. We further propose and motivate two properties that neuro-symbolic methods should ideally possess.

[1]This is an extended version of the position paper published at NeSy'19 [De Raedt et al., 2019].

2 DeepProbLog

One example of a neuro-symbolic integration system is DeepProbLog [Manhaeve et al., 2018], which is an extension of the probabilistic programming language ProbLog [Fierens et al., 2015]. We will first briefly explain how inference in ProbLog can be considered a weighted model counting problem, and how this allows extending it towards DeepProbLog.

Probabilistic inference as Weighted Model Counting The underlying concept of a probabilistic logic programming language is simple: (ground) atomic expressions of the form $q(t_1, ..., t_n)$ are considered as independent random variables that have a probability p of being true. This view, which was pioneered in the distribution semantics by Sato [1995], effectively unifies the basic primitives in logic with those in probability theory: propositions become random variables. On top of these "probabilistic facts" one can then define rules (in the form of logic programs) that allow to derive conclusions from these facts and induce a probability distribution over possible worlds. Inference in ProbLog can be viewed as a weighted model counting (WMC) problem, namely, the probability of a query q in a logic program that contains a set of probabilistic facts $\mathcal{F}$ is defined as:

$$P(q) = \sum_{F \subseteq \mathcal{F} : q \in w_F} \prod_{f_i \in F} p_i \prod_{f_i \in \mathcal{F} \setminus F} (1 - p_i)$$

where p_i is the probability associated with the fact f_i, and w_F is the set that is the union of $F \subseteq \mathcal{F}$ and the set of ground atoms that are entailed by F and the rules in the logic program.

Neuro-symbolic integration as Algebraic Model Counting DeepProbLog integrate neural networks by extending the idea of the probabilistic fact to that of the neural predicate. The neural predicate allow atomic expressions to be annotated with a functor that represents the output of a neural network, given that it can be considered a probability. This simple idea is appealing as it allows us to retain all the essential components of the probabilistic logic programming language: the semantics, the inference mechanism, as well as the implementation. At the same time, it clearly decouples the logical component from the neural component. As neural networks become predicates, they can be called from the logic programs and so the logical layer is at the higher-level and the neural one at the lower-level with the neural predicates providing the interface, while both sides can treat each other as a black box.

To allow neural networks to be trained with gradient-descent based methods, DeepProbLog has to be able to derive the gradients that are used to perform backpropagation in the neural networks. This is done by casting it as an algebraic model counting (AMC) problem [Kimmig et al., 2017]. The algebraic model count of a query in a logic program is defined as

$$A(q) = \bigoplus_{F \subseteq \mathcal{F} : q \in w_F} \bigotimes_{f_i \in F} l(f_i) \bigotimes_{f_i \in \mathcal{F} \setminus F} l(\neg p_i)$$

where $\bigotimes$ and $\bigoplus$ are the operators of a commutative semiring, and $l(f)$ a function that maps facts in F onto elements of that semiring. It is clear that WMC can be considered a special case of AMC with the probability semiring. Another useful semiring is the gradient semiring [Eisner, 2002], which enables automatic differentiation alongside the standard ProbLog inference.

Example Consider the task where we have to learn to classify sums from pairs of handwritten digits, e.g $+$ $= 8$. This task can be easily solved in DeepProbLog by specifying a neural predicate that defines the classification of single digits, and a single line of logic that defines the addition as follows:

```
nn(classifier_id, [X], Y, [0 .. 9]) :: digit(X,Y).
addition(X,Y,Z) :-digit(X,N1), digit(Y,N2), Z is N1+N2.
```

As shown in Manhaeve et al. [2018], this single line of logic is enough to clearly outperform a fully neural baseline, as it converges quicker, to a higher accuracy. Furthermore, because the neural network in the DeepProbLog program classifies digits, it can be reused for other tasks after training, whereas the baseline can only be used on the same task. The computation of the algebraic model count that produces the probability and the gradients w.r.t. the output of the neural network is shown in Figure 1. In describing the learning algorithm for DeepProbLog it is important to note that DeepProbLog, in

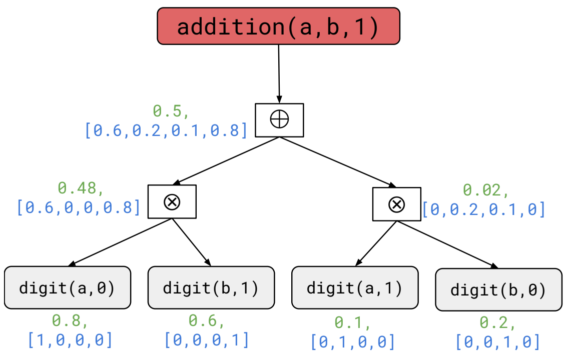

Figure 1: The computation of probabilities(green) and gradients(blue) using algebraic model counting.

contrast to other approaches for neuro-symbolic learning, not only encompasses a neural and a logical framework, but also a probabilistic one. Making the logic probabilistic also makes it differentiable, allowing gradients to be derived so that the neural networks can be trained with standard gradient descent methods. For the given example, note that the neural predicate is trained from an indirect signal, i.e., from observations of sums (at the output of the logical layer), rather than through labels of individual digits.

3 Position statement

Based on our experience in upgrading machine learning systems towards the use of (probabilistic) logical and relational representations [De Raedt, 2008, Muggleton et al., 2012], we argue that a first desirable property of frameworks that integrate two other frameworks A and B, is to have the original frameworks A and B as a special case of the integrated one. If A or B cannot be fully reconstructed, one clearly loses certain abilities, which is not only undesirable but which also implies that there is no true unification. Applying this property to neuro-symbolic computation implies that the existing frameworks should have both the neural and the symbolic representations as special case. When this is the case, one retains both the learning abilities of the neural component as well as the reasoning and learning abilities and the semantics of the symbolic representations. Unfortunately, this property is not satisfied by the vast majority of neuro-symbolic approaches.

We also advocate a second desirable property for neuro-symbolic computation: models that learn from observed samples should be able to deal with uncertainty. Therefore, one should not only integrate logic with neural networks in neuro-symbolic computation, but also probability. This effectively leads to an integration of probabilistic logics (hence statistical relational AI) with neural networks and opens up new abilities. Furthermore, although at first sight, this may appear as a complication, it actually can greatly simplify the integration of neural networks with logic.

DeepProbLog achieves both of these properties. It is easy to see how both the logic and the neural networks are preserved as special cases: if the DeepProbLog program has no neural predicates, it simply becomes a normal ProbLog program. If that program has no probabilistic facts, it becomes a pure logic program. Furthermore, if the program only contains a neural predicate, it becomes equivalent to directly training the neural network.

4 State-of-the-art in Neuro-Symbolic Computation

We now examine the state-of-the-art in neuro-symbolic computation with respect to these two desired properties. Here, we only consider works that combine neural networks with logic, although the field of neuro-symbolic integration is wider than this. The results are summarized in Table 1. This subset of the domain can be divided into two categories: logic as constraints and differentiable logic frameworks.

Logic as constraints. The dominant idea is to use logic as a constraint on a deep model. That is, the deep model is extended with a regularization term, derived from the desired logical properties, which encourages it to mimic logical reasoning.For instance, Diligenti et al. [2017] and Donadello

Table 1: Comparing neuro-symbolic frameworks. 1a and 1b refer to keeping neural networks and logic respectively as edge cases. 2 refers to having probabilities.

	1a	1b	2	Underlying logic
Semantic based regularization [Diligenti et al., 2017]	X			FOL
Logic Tensor Networks [Donadello et al., 2017]	X			FOL
Lifted Rule Injection [Demeester et al., 2016]	X			Implication rules
Adverserial Set Regularisation [Minervini et al., 2017]	X			Clausal logic
Semantic Loss Function [Xu et al., 2018]	X		X	Propositional logic
∂ILP [Evans and Grefenstette, 2018]				Datalog
Neural Theorem Prover [Rocktäschel and Riedel, 2017]				Clausal logic
TensorLog [Cohen et al., 2018]		X		Datalog Subset
DeepProbLog [Manhaeve et al., 2018]	X	X	X	Clausal logic

et al. [2017] use first-order logic to specify constraints on the output of the neural network, while Demeester et al. [2016] and Minervini et al. [2017] use logical `IF-THEN` rules, derived from expert knowledge, to enforce the embeddings to be more consistent with the logical constraints. Xu et al. [2018] introduce a generalization of this principle such that more complex logical constraints can be imposed on any deep model. Common to all these approaches is the use of logic as a regularizer, which encourages deep models to satisfy the constraints. This leads to the logic being encoded into the parameters (either into the weights of the neural network, or directly into the embeddings). The constraints are soft, and there is no guarantee that they all will be satisfied. In fact, the trade-off between encouraging the constraints vs. following the data in case both are contradictory, is merely a hyperparameter of the model. These methods thus generally do have neural networks as a special case (i.e., when there are no additional constraints), however, they do not have logic as a special case when leaving out the neural part. The importance of being able to fully recover the logic is already hinted at in some of these papers. For example, Xu et al. [2018] show that neural networks trained with the additional regularization term cannot consistently make predictions coherent with the logic they were trained on.

Differentiable logic frameworks. A different class of neuro-symbolic systems works by making the logic program differentiable, which is achieved by reformulating the basic reasoning primitives using the mathematics of differentiable functions. Rocktäschel and Riedel [2017] apply this to the backward reasoning procedure of Prolog, while Evans and Grefenstette [2018] do this in the style of forward reasoning. Although both systems are based on standard reasoning algorithms, the original logic cannot be recovered as special case, since the original semantics and inference have been fundamentally changed. Similarly, Cohen et al. [2018] introduce a framework to compile a tractable subset of logic programs into differentiable functions and to execute it with neural networks. All three systems mentioned above construct a neural network to perform the execution, but they do not have neural networks as a special case. These approaches can only deal with rather restricted classes of logic programs. Furthermore, because they push the logic programming components into the neural networks, both the proper semantics of logic programs and the programming language nature of the framework are sacrificed. These systems also suffer from not keeping the original logic as a special case. For example, Rocktäschel and Riedel [2017] and Evans and Grefenstette [2018] both mention that the neural execution of the differentiable logic creates a large computational overhead.

5 Conclusion

In this paper, we showed how we integrated logic and learning in the DeepProbLog system through the use of algebraic model counting. Furthermore, we argued that neuro-symbolic integration systems should have two desirable properties: to preserve logic and neural networks as edge cases, and to allow integrating a probabilistic framework alongside the neural and logical one.

Acknowledgments

This work has been partially supported by the European Research Council Advanced Grant project SYNTH9 (ERCAdG-694980). RM is a SB PhD fellow at FWO (1S61718N).

References

William W Cohen, Fan Yang, and Kathryn Rivard Mazaitis. Tensorlog: Deep learning meets probabilistic databases. *Journal of Artificial Intelligence Research*, 1:1–15, 2018.

Luc De Raedt. *Logical and Relational Learning*. Springer, 2008.

Luc De Raedt and Angelika Kimmig. Probabilistic (logic) programming concepts. *Machine Learning*, 100(1):1–43, 2015.

Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. Statistical relational artificial intelligence: Logic, probability and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2016.

Luc De Raedt, Robin Manhaeve, Sebastijan Dumancic, Thomas Demeester, and Angelika Kimmig. Neuro-symbolic= neural+ logical+ probabilistic. In *NeSy'19@ IJCAI, the 14th International Workshop on Neural-Symbolic Learning and Reasoning*, pages 1–4, 2019.

Thomas Demeester, T. Rocktäschel, and S. Riedel. Lifted rule injection for relation embeddings. In *EMNLP*, 2016.

Michelangelo Diligenti, Marco Gori, and Claudio Sacca. Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165, 2017.

Ivan Donadello, Luciano Serafini, and Artur S. d'Avila Garcez. Logic tensor networks for semantic image interpretation. In *IJCAI*, 2017.

Jason Eisner. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th annual meeting on Association for Computational Linguistics*, pages 1–8. Association for Computational Linguistics, 2002.

Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *JAIR*, 2018.

Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory and Practice of Logic Programming*, 15(3):358–401, 2015.

Artur d'Avila Garcez, Tarek R Besold, Luc De Raedt, Peter Földiak, Pascal Hitzler, Thomas Icard, Kai-Uwe Kühnberger, Luis C Lamb, Risto Miikkulainen, and Daniel L Silver. Neural-symbolic learning and reasoning: contributions and challenges. In *2015 AAAI Spring Symposium Series*, 2015.

Artur S d'Avila Garcez, Krysia B Broda, and Dov M Gabbay. *Neural-symbolic learning systems: foundations and applications*. Springer Science & Business Media, 2012.

Lise Getoor and Ben Taskar. *Introduction to statistical relational learning*. MIT press, 2007.

Barbara Hammer and Pascal Hitzler. *Perspectives of neural-symbolic integration*, volume 8. Springer Heidelberg:, 2007.

Daniel Kahneman. *Thinking, fast and slow*. Farrar, Straus and Giroux New York, 2011.

Angelika Kimmig, Guy Van den Broeck, and Luc De Raedt. Algebraic model counting. *Journal of Applied Logic*, 22:46–62, 2017.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In *NeurIPS*, 2018.

Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Adversarial sets for regularising neural link predictors. In *UAI*, 2017.

Stephen Muggleton, Luc De Raedt, David Poole, Ivan Bratko, Peter Flach, Katsumi Inoue, and Ashwin Srinivasan. Ilp turns 20. *Machine learning*, 86(1):3–23, 2012.

Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In *NIPS*, 2017.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. A simple neural network module for relational reasoning. In *NIPS*, 2017.

T. Sato. A statistical learning method for logic programs with distribution semantics. In *ICLP*, 1995.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *ICML*, 2018.