DeepProbLog: Neural Probabilistic Logic Programming

Robin Manhaeve[1], Sebastijan Dumančić[1], Angelika Kimmig[2], Thomas Demeester[3], and Luc De Raedt[1]

[1] KU Leuven
firstname.lastname@cs.kuleuven.be
[2] Cardiff University
KimmigA@cardiff.ac.uk
[3] Ghent University - imec
thomas.demeester@ugent.be

Abstract. Joining the full flexibility of high-level probabilistic reasoning with the representational power of deep neural networks is still an open problem. With DeepProbLog [4], we start from Problog [2], a probabilistic logic programming language (PLP) and extend it with neural predicates. The neural predicate represents the relation between the input and output as defined by a neural network. It allows us to integrate neural networks into ProbLog in a way that retains its semantics and most of its inference. We demonstrate the capabilities of DeepProbLog in combined symbolic and subsymbolic reasoning, program induction, and probabilistic logic programming. This work is published at NeurIPS 2018.

1 DeepProbLog

The Neural Predicate ProbLog lifts Prolog to a PLP by allowing facts to be annotated with probabilities. Similarly, DeepProbLog integrates neural networks by allowing facts to be annotated with a special functor that represent a neural network. These neural networks can be considered functions that, when ground, return a probability distribution. These facts (neural predicates) can be integrated in standard ProbLog inference by replacing the functor with the corresponding probability distribution, turning them into regular probabilistic facts.

Learning Apart from including neural networks in inference, DeepProbLog is also able to train the neural networks represented by the neural predicates. We use the *learning from entailment* setting. In contrast to the earlier approach for ProbLog parameter learning in this setting, we use gradient descent rather than EM, as this allows for seamless integration with neural network training. More specifically, to compute the gradient with respect to the probabilistic logic program part, we rely on aProbLog [3], a generalization of the ProbLog language and its inference to arbitrary commutative semirings, including the gradient semiring. This semiring allows us to perform gradient derivation in parallel with the inference. The resulting gradients are used to update the probabilistic parameters in the logic program. Additionally, the gradients derived for the neural predicates are used to start backpropagation in the neural network, which derives the gradients for the internal parameters. Then, standard gradient-based optimizers are used to update the parameters of the network.

2 Experiments

MNIST addition To show that DeepProbLog supports both logical reasoning and deep learning, we extend the classic learning task on the MNIST dataset to a task that requires reasoning. We divide the MNIST dataset into pairs of images and label each pair with their sum. We compare a simple DeepProbLog model that encodes this addition in logic to a CNN baseline. We show that the DeepProbLog model trains faster and achieves a higher a final accuracy.

Program Induction The second set of experiments demonstrate that DeepProbLog can achieve a form of program induction. We follow the program sketch setting of $\partial 4$ [1], where holes in programs are filled by neural networks. These are trained using examples that define the input and output for the entire program. We consider three tasks: addition, sorting and word algebra problems (WAPs). DeepProbLog achieves the same performance as $\partial 4$ on all experiments except for the sorting problem. When training on lists larger than 3 elements, $\partial 4$ was unable to converge due to computational issues [1] but DeepProbLog still achieves 100% accuracy.

Probabilistic programming and deep learning In the final experiment we demonstrate that DeepProbLog can simultaneously perform probabilistic reasoning, probabilistic learning and neural learning. To do this, we design an experiment in which a lottery game is played that involves training two separate neural networks and learning the probabilistic parameters of the program. We show that DeepProbLog is able to achieve 100% accuracy, training both neural networks and learning the correct probabilistic parameters.

References

1. Bošnjak, M., Rocktäschel, T., Riedel, S.: Programming with a differentiable forth interpreter. In: ICML. vol. 70, pp. 547–556 (2017)
2. Fierens, D., Van den Broeck, G., Renkens, J., Shterionov, D., Gutmann, B., Thon, I., Janssens, G., De Raedt, L.: Inference and learning in probabilistic logic programs using weighted Boolean formulas. Theory and Practice of Logic Programming **15**(3), 358–401 (2015)
3. Kimmig, A., Van den Broeck, G., De Raedt, L.: An algebraic Prolog for reasoning about possible worlds. In: AAAI (2011)
4. Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., De Raedt, L.: Deepproblog: Neural probabilistic logic programming. In: NeurIPS. pp. 3749–3759 (2018)