Inference in Relational Neural Machines[1]

Giuseppe Marra[2] and **Michelangelo Diligenti**[3] and **Lapo Faggi**[2] and **Francesco Giannini**[2] and **Marco Gori**[2] and **Marco Maggini**[2]

Abstract. Integrating logic reasoning and deep learning from sensory data is a key challenge to develop artificial agents able to operate in complex environments. Whereas deep learning can operate at a large scale thanks to recent hardware advancements (GPUs) as well as other important technical advancements like Stochastic Gradient Descent, logic inference can not be executed over large reasoning tasks, as it requires to consider a combinatorial number of possible assignments. Relational Neural Machines (RNMs) have been recently introduced in order to co-train a deep learning machine and a first-order probabilistic logic reasoner in a fully integrated way. In this context, it is crucial to avoid the logic inference to become a bottleneck, preventing the application of the methodology to large scale learning tasks. This paper proposes and compares different inference schemata for Relational Neural Machines together with some preliminary results to show the effectiveness of the proposed methodologies.

1 Introduction

Empowering machine learning with explicit reasoning capabilities is a key step toward a trustworthy and human-centric AI, where the decisions of the learners are explainable and with human-understandable guarantees. While sub-symbolic approaches like deep neural networks have achieved impressive results in several tasks [7], standard neural networks can struggle to represent relational knowledge on different input patterns or relevant output structures. Recently, some work has been done to learn and inject relational features into the learning process [17, 16]. Symbolic approaches [3, 18] based on probabilistic logic reasoners can perform an explicit inference process in presence of uncertainty. Another related line of research studies hybrid approaches leveraging deep learning schemas and neural networks to learn the structure of the reasoning process like done, for instance, by Neural Theorem Provers [19] or TensorLog [11].

However, bridging the gap between symbolic and sub-symbolic levels is still an open problem which has been recently addressed by neuro-symbolic approaches [12, 20]. Hu et al. [10] inject the prior knowledge into the network weights via a distillation process but with no guarantee that the logic will be properly generalized to the test cases. Deep Structured Models [1] and Hazan et al. [9] imposes statistical structure on the output predictions. The Semantic Loss [22] defines a loss which encodes the desired output structure. However, the loss does not define a probabilistic reasoning process, limiting the flexibility of the approach. Deep ProbLog [13] extends the probabilistic logic programming language ProbLog [3] with predicates implemented by a deep learner. This approach is powerful but limited to cases where exact inference is possible. Deep Logic Models [14] improve over related approaches like Semantic-based Regularization [4], and Logic Tensor Networks [5], but they fail to perfectly match the discrimination abilities of a pure-supervised learner.

Relational Neural Machines (RNM) [15] can perfectly replicate the effectiveness of training from supervised data of deep architectures, while integrating the full expressivity and rich reasoning process of Markov Logic Networks [18]. RNMs like any probabilistic reasoner based on a graphical model representing the fully grounded FOL knowledge are strongly limited in the scale at which they can operate. Indeed, the large combinatorial number of possible assignments together with the complex casual dependency structure of the inference requires to device appropriate approximate inference algorithmic solutions. This paper proposes and studies different new inference solutions that are thought to be effective for RNMs.

The outline of the paper is as follows. Section 2 presents the model and how it can be used to integrate logic and learning. Sections 3 and 4 study tractable approaches to perform training and inference with the model, respectively. Section 5 shows the experimental evaluation of the proposed ideas on various datasets. Finally, Section 6 draws some conclusions and highlights some planned future work.

2 Model

A Relational Neural Machine establishes a probability distribution over a set of n output variables of interest $\boldsymbol{y} = \{y_1, \ldots, y_n\}$, given a set of predictions made by one or multiple deep architectures, and the model parameters. In this paper the output variables are assumed to be binary, i.e. $y_i = \{0, 1\}$.

Unlike standard neural networks which compute the output via a simple forward pass, the output computation in an RNM can be decomposed into two stages: a *low-level* stage processing the input patterns, and a subsequent *semantic* stage, expressing constraints over the output and performing higher level reasoning. The first stage processes D input patterns $\boldsymbol{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_D\}$, returning the values $\boldsymbol{f}$ using the network with parameters $\boldsymbol{w}$. The higher layer takes as input $\boldsymbol{f}$ and applies reasoning using a set of constraints, whose parameters are indicated as $\boldsymbol{\lambda}$, then it returns the set of output variables $\boldsymbol{y}$.

A RNM model defines a conditional probability distribution in the exponential family defined as:

$$p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\lambda}) = \frac{1}{Z} \exp\left(\sum_c \lambda_c \Phi_c(\boldsymbol{f}, \boldsymbol{y})\right)$$

where Z is the partition function and the C potentials Φ_c express some properties on the input and output variables. The parameters

[2] Department of Computer Science, KU Leuven, Belgium
[3] Department of Information Engineering and Science, University of Siena, Italy

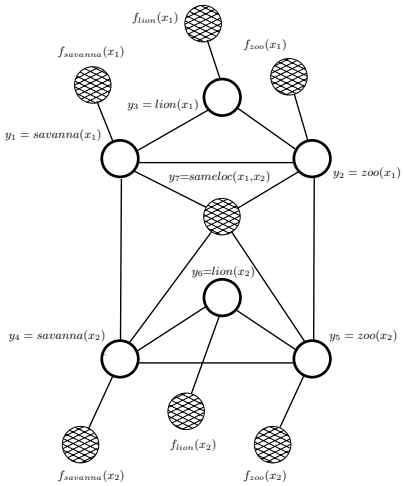

Figure 1. The graphical model representing an RNM, where the output variables $\boldsymbol{y}$ depend on the output of first stage $\boldsymbol{f}$, processing the inputs $\{x_1, x_2\}$ instantiated for the rules $\forall x\ lion(x) \Rightarrow savanna(x) \vee zoo(x)$, $\forall x \forall x'\ sameloc(x, x') \wedge savanna(x) \Rightarrow savanna(x')$, and $\forall x \forall x'\ sameloc(x, x') \wedge zoo(x) \Rightarrow zoo(x')$.

$\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_C\}$ determine the strength of the potentials Φ_c.

In a classical and pure supervised learning setup, the patterns are i.i.d., it is therefore possible to split the $\boldsymbol{y}$, $\boldsymbol{f}$ into disjoint sets grouping the variables of each pattern, forming separate cliques. Let us indicate as $\boldsymbol{y}(x)$, $\boldsymbol{f}(x)$ the portion of the output and function variables referring to the processing of an input pattern x. A single potential Φ_0 corresponding to the dot product between $\boldsymbol{y}$ and $\boldsymbol{f}$ is needed to represent supervised learning, and this potential decomposes over the patterns yielding the distribution,

$$p_0(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\lambda}) = \frac{1}{Z} \exp\left(\sum_{x \in S} \phi_0(\boldsymbol{y}(x), \boldsymbol{f}(x))\right) \tag{1}$$

where $S \subseteq \boldsymbol{x}$ is the set of supervised patterns. As shown by Marra et al. [15], the cross entropy loss over sigmoidal or softmax outputs can be exactly recovered by maximizing the log-likelihood of a RNM for one-label and multi-label classification tasks, respectively.

Neuro-symbolic integration can be obtained by employing one potential Φ_0 enforcing the consistency with the supervised data together with potentials representing the logic knowledge. Using a similar approach to Markov Logic Networks, a set of First–Order Logic (FOL) formulas is input to the system, and a potential Φ_c for each formula is considered. It is assumed that some (or all) the predicates in a KB are unknown and need to be learned together with the parameters driving the reasoning process.

In the following we refer to *grounded* expression (the same applies to atom or predicate) as a FOL rule whose variables are assigned to specific constants. It is assumed that the undirected graphical model is built such that: each grounded atom corresponds to a node in the graph; all the nodes corresponding to grounded atoms co-occurring in at least one rule are connected on the graph. As a result, there is one clique (and then potential) for each grounding g_c of the formula in $\boldsymbol{y}$. It is assumed that all the potentials resulting from the c-th formula share the same weight λ_c, therefore the potential Φ_c is the sum over all groundings of ϕ_c in the world $\boldsymbol{y}$, such that: $\Phi_c(\boldsymbol{y}) = \sum_{\boldsymbol{y}_{c,g}} \phi_c(\boldsymbol{y}_{c.g})$ where $\phi_c(g_c)$ assumes a value equal to 1 and 0 if the grounded formula holds true and false. This yields the probability distribution:

$$p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\lambda}) = \frac{1}{Z} \exp\left(\sum_x \phi_0(\boldsymbol{f}(x), \boldsymbol{y}(x)) + \sum_c \lambda_c \sum_{\boldsymbol{y}_{c,g}} \phi_c(\boldsymbol{y}_{c,g})\right)$$

This will allow to develop the data embeddings as part of training by enforcing the consistency between the reasoner and network outputs, while distilling the logical knowledge into the network weights.

Figure 1 shows the graphical model obtained for a simple multi-class image classification task. The goal of the training process is to train the classifiers approximating the predicates, but also to establish the relevance of each rule. For example, in an image classification task, the formula $\forall x\ Antelope(x) \wedge Lion(x)$ is likely to be associated to a higher weight than $\forall x\ PolarBear(x) \wedge Lion(x)$, which are unlikely to correlate in the data.

3 Training

The computation of the partition function requires a summation over all possible assignments of the output variables, which is intractable for all but trivial cases. A particularly interesting case is when it is assumed that the partition function factorizes over the potentials like done in piecewise likelihood [21]:

$$Z \approx \prod_c Z_c = \prod_c \left[\sum_{\boldsymbol{y}'_c} \exp(\lambda_c \Phi_c(\boldsymbol{f}, \boldsymbol{y}'_c))\right] \tag{2}$$

where $\boldsymbol{y}_c$ is the subset of variables in $\boldsymbol{y}$ that are involved in the computation of Φ_c. Then the piecewise-local probability for the c-th constraint can be expressed as:

$$p_c^{PL}(\boldsymbol{y}_c|\boldsymbol{f}, \boldsymbol{\lambda}) = \frac{\exp\left(\lambda_c \Phi_c(\boldsymbol{f}, \boldsymbol{y}_c)\right)}{Z_c}$$

Under this assumption, the factors can be distributed over the potential giving the following generalized piecewise likelihood:

$$p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\lambda}) \quad \approx \quad \prod_c p(\boldsymbol{y}_c|\boldsymbol{y} \backslash \boldsymbol{y}_c, \boldsymbol{f}, \lambda_c) = \prod_c p_c^{PL}(\boldsymbol{y}_c|\boldsymbol{f}, \boldsymbol{\lambda})$$

If the variables in $\boldsymbol{y}$ are binary, the computation of Z requires summation over all possible assignments which has $O(2^{|\boldsymbol{y}|})$ complexity. Using the local decomposition this is reduced to $O(|\boldsymbol{y}_{\bar{c}}| \cdot 2^{n_{\bar{c}}})$, where $\bar{c}$ is the index of the formula corresponding to the potential with the largest number $n_{\bar{c}}$ of variables to ground.

If the c-th constraint is factorized using the PL partitioning, the derivatives of the log-likelihood with respect to the model potential weights are:

$$\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\lambda})}{\partial \lambda_c} \approx \Phi_c(\boldsymbol{f}, \boldsymbol{y}) - E_{p_c^{PL}}[\Phi_c]$$

and with respect to the learner parameters:

$$\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\lambda})}{\partial w_k} \approx \sum_i \frac{\partial f_i}{\partial w_k} \left[y_i - E_{y' \sim p_0^{PL}}\left[y'_i\right]\right]$$

EM When the world is not fully observed during training, an iterative Expectation Maximization (EM) schema can be used to marginalize over the unobserved data in the expectation step using the inference methodology as described in the next paragraph. Then, the average constraint satisfaction can be recomputed, and, finally, the $\boldsymbol{\lambda}, \boldsymbol{w}$ parameters can be updated in the maximization step. This process is then iterated until convergence.

4 Inference.

This sections proposes some general methodologies which can be used to make RNM inference tractable.

Inference tasks can be sub-categorized into different groups. In particular, MAP inference methods search the most probable assignment of the $\boldsymbol{y}$ given the evidence and the fixed parameters $\boldsymbol{w}, \boldsymbol{\lambda}$. The problem of finding the best assignment $\boldsymbol{y}^\star$ to the unobserved query variables given the evidence $\boldsymbol{y}^e$ and current parameters can be stated as:

$$\boldsymbol{y}^\star = arg\max_{\boldsymbol{y}'} \sum_c \lambda_c \Phi_c(\boldsymbol{f}, [\boldsymbol{y}', \boldsymbol{y}^e]) \tag{3}$$

where $[\boldsymbol{y}', \boldsymbol{y}^e]$ indicates a full assignment to the $\boldsymbol{y}$ variables, split into the query and evidence sets.

On the other hand, MARG inference methods compute the marginal probability of a set of random variables given some evidence. MARG inference sums the probability of an assignment of a query variable y_q over all possible worlds. MARG inference for a single query variable is defined as:

$$p(y_q|\boldsymbol{\lambda}, \boldsymbol{f}, \boldsymbol{y}^e) = \sum_{\boldsymbol{y}\setminus\{\boldsymbol{y}^e, y_q\}} p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\lambda}, \boldsymbol{y}^e) \quad \forall y_q \in \boldsymbol{y}$$

Both MAP and MARG inference are intractable in the most general cases as they require to consider all possible assignments. Therefore, approximate methods must be devised to be able to tackle the most interesting applications. This section proposes a few inference solutions that can be naturally applied to RNM.

Piecewise MAP Inference. MAP inference in RNMs requires to evaluates all possible $2^{|\boldsymbol{y}|}$ assignments, which is generally intractable. A possible solution is to employ the the piecewise approximation (Equation 2) to separately optimize each single factor, so reducing the complexity to $2^{n_{\hat{c}}}$ with $n_{\hat{c}}$ the size of the largest factor. The main issue with this solution is that the same variable can be present in different factors, and the piecewise assignments can be inconsistent across the factors. The assignments to variables shared across factors can be performed by selecting the assignment selected by the most factors.

Fuzzy Logic MAP Inference. The $\boldsymbol{y}$ values can be relaxed into the $[0, 1]$ interval and assume that each potential $\Phi_c(\boldsymbol{f}, \boldsymbol{y})$ has a fuzzy-logic [8] continuous surrogate $\Phi_c^s(\boldsymbol{f}, \boldsymbol{y})$ which collapses into the original potential when the $\boldsymbol{y}$ assume crisp values and is continuous with respect to each y_i. When relaxing the potentials to accept continuous variables, the MAP problem stated by Equation 3 can be solved by gradient-based techniques, by computing the derivative with respect of each output variable:

$$\frac{\partial \log p(\boldsymbol{y}'|\boldsymbol{y}^e, \boldsymbol{f}, \boldsymbol{\lambda})}{\partial y_k} = f_k + \sum_c \lambda_c \frac{\partial \Phi_c^s(\boldsymbol{f}, [\boldsymbol{y}', \boldsymbol{y}^e])}{\partial y_k}$$

Different t-norm fuzzy logics have been considered as continuous relaxations of logic formulas [4] e.g. Gödel, Łukasiewicz and Product logics. Furthermore, a fragment of the Łukasiewicz logic [6] has been recently proposed for translating logic inference into a convex optimization problem.

Piecewise MARG. The piecewise approximation defined by Equation 2 allows to marginalize the probability of an assignment over the single factors, allowing to efficiently perform MARG inference as:

$$p(y_q|\boldsymbol{\lambda}, \boldsymbol{f}, \boldsymbol{y}^e) \approx \sum_{\boldsymbol{y}_u = \boldsymbol{y}\setminus(\boldsymbol{y}^e, y_q)} \Big[p_0^{PL}(\boldsymbol{y}_u, y_q|\boldsymbol{f}) \cdot \cdot \prod_c p_c^{PL}(\boldsymbol{y}_u, y_q|\lambda_c, \boldsymbol{f}, \boldsymbol{y}^e) \Big]$$

Finally the shared variable can be reconciled by selecting the assignment for a shared variable that has the highest marginal probability.

Piecewise Gibbs. Gibbs sampling can be used to sample from the distribution and then select the sample with the highest probability:

$$\boldsymbol{y}^\star = arg\max_{\boldsymbol{y}' \sim p} \sum_c \lambda_c \Phi_c(\boldsymbol{f}, [\boldsymbol{y}', \boldsymbol{y}^e])$$

In order to speed up the process, a blocked Gibbs sampler may be considered, by grouping many variables and then sampling by their joint distribution. For instance, piecewise approaches suggest to group the variables belonging to potential, exploiting the constraints expressed by the potential on the samples. To speed up the process, a certain flip can be accepted, only if it yields a strictly greater probability value (Monotonic Gibbs sampling).

Piecewise Gibbs with Fuzzy Map Inference. Gibbs sampling would generally require a high number of samples to converge to the correct distribution. Hybrid inference methodologies can be used to reduce the burn in time by starting the Gibbs sampler from the MAP solution found using efficient approximate inference methods like Fuzzy Logic MAP. The sampler then modifies the solution by iteratively sampling from the piecewise local distributions. Combining Fuzzy Logic MAP and a Gibbs sampler allows to avoid low-quality local minima where fuzzy MAP solutions can get stuck, while speeding up the Gibbs sampling convergence.

Relax, Compensate and Recover. Relax, Compensate & Recover (RCR) [2] is an algorithm to transform a graphical model into a simplified model, where inference is tractable. This simplified model is changed while running the algorithm, by computing compensations to recover a solution as close as possible to the correct distribution.

Graph Neural Networks. A few recent works show that inference in probabilistic models can be approximated by using Graph Neural Networks [23]. This would allow to define an end-to-end neural RNM formulation.

5 Experiments

The experimental evaluation aims at testing some of the proposed inference methods. The evaluation is still partial as more methods are currently being implemented.

The evaluation is carried out on a toy task, that is designed to highlight the capability of RNMs to learn and employ soft rules that are holding only for a sub-portion of the whole dataset. The MNIST dataset contains images of single handwritten digits. In this task it is assumed that additional relational logic knowledge is available in the form of a binary predicate $link$ connecting image pairs. Given two images x, y, whose corresponding digits are denoted by i, j, a link between x and y is established if the second digit follows the first one, i.e. $i = j + 1$. However, the $link$ predicate can be noisy,

such that there is a degree of probability that the $link(x, y)$ is established for $i \neq j + 1$. The knowledge about the $link$ predicate can be represented by the FOL formulas:

$$\forall x \forall y \; link(x, y) \wedge digit(x, i) \Rightarrow digit(x, i + 1) \quad i = 0, \ldots, 8 \, ,$$

where $digit(x, i)$ is a binary predicate indicating if a number i is the digit class of the image x. Since the $link$ predicate holds true also for pairs of non-consecutive digits, the above rule is violated by a certain percentage of digit pairs. Therefore, the manifolds established by the formulas can help in driving the predictions, but the noisy links force the reasoner to be flexible about how to employ the knowledge. The training set is created by randomly selecting 1000 images from the MNIST dataset and by adding the $link$ relation with an incremental degree of noise. For each degree of noise in the training set, we created an equally sized test set with the same degree of noise. A neural network with 100 hidden relu neurons is used to process the input images. Table 1 reports the results of RNM inference methods against the baseline provided by the neural network varying the percentage of links that are predictive of a digit to follow another one. Using an EM based schema tends to consistently outperform other methods, this is because EM allows to also improve the underlying neural network by back-propagating the reasoning predictions to the learner. Other inference methods will be tested within EM. Fuzzy MAP tends to find good solutions constantly improving over the baseline. It is important to notice how RNM is robust with respect to the high link noise levels even when the relational data is not carrying any useful information, the final solution still matches the baseline.

%link noise	NN	Fuzzy MAP	Piecewise Gibbs	Fuzzy MAP EM
0.0	0.78	1.00	1.00	1.00
0.2	0.78	1.00	1.00	1.00
0.4	0.78	0.99	0.98	0.96
0.6	0.78	0.89	0.88	0.96
0.8	0.78	0.86	0.64	0.86
0.9	0.78	0.78	0.28	0.78

Table 1. Accuracy of the different inference methods with respect to the percentage of link that are established between digit image pairs violating the logical rule.

6 Conclusions and Future Work

This paper shows different inference methods for Relational Neural Machines, a novel framework to provide a tight integration between learning from supervised data and logic reasoning. As future work, we plan to undertake a larger experimental exploration of RNM for more structured problems using all the inference methods proposed by this paper.

ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 825619.

REFERENCES

[1] Liang-Chieh Chen, Alexander Schwing, Alan Yuille, and Raquel Urtasun, 'Learning deep structured models', in *International Conference on Machine Learning*, pp. 1785–1794, (2015).

[2] Arthur Choi and Adnan Darwiche, 'Relax, compensate and then recover', in *JSAI International Symposium on Artificial Intelligence*, pp. 167–180. Springer, (2010).

[3] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen, 'Problog: A probabilistic prolog and its application in link discovery', in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pp. 2468–2473, San Francisco, CA, USA, (2007). Morgan Kaufmann Publishers Inc.

[4] Michelangelo Diligenti, Marco Gori, and Claudio Sacca, 'Semantic-based regularization for learning and inference', *Artificial Intelligence*, **244**, 143–165, (2017).

[5] I Donadello, L Serafini, and AS d'Avila Garcez, 'Logic tensor networks for semantic image interpretation', in *IJCAI International Joint Conference on Artificial Intelligence*, pp. 1596–1602, (2017).

[6] Francesco Giannini, Michelangelo Diligenti, Marco Gori, and Marco Maggini, 'On a convex logic fragment for learning and reasoning', *IEEE Transactions on Fuzzy Systems*, (2018).

[7] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio, *Deep learning*, volume 1, MIT press Cambridge, 2016.

[8] Petr Hájek, *Metamathematics of fuzzy logic*, volume 4, Springer Science & Business Media, 2013.

[9] Tamir Hazan, Alexander G Schwing, and Raquel Urtasun, 'Blending learning and inference in conditional random fields', *The Journal of Machine Learning Research*, **17**(1), 8305–8329, (2016).

[10] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard H. Hovy, and Eric P. Xing, 'Harnessing deep neural networks with logic rules', in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, (2016).

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

[12] Navdeep Kaur, Gautam Kunapuli, Tushar Khot, Kristian Kersting, William Cohen, and Sriraam Natarajan, 'Relational restricted boltzmann machines: A probabilistic logic learning approach', in *International Conference on Inductive Logic Programming*, pp. 94–111. Springer, (2017).

[13] Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt, 'Deepproblog: Neural probabilistic logic programming', *arXiv preprint arXiv:1805.10872*, (2018).

[14] Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, and Marco Gori, 'Integrating learning and reasoning with deep logic models', in *Proceedings of the European Conference on Machine Learning*, (2019).

[15] Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Marco Maggini, and Gori. Marco, 'Reational neural machines', in *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, (2020).

[16] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio, 'Holographic embeddings of knowledge graphs', in *Thirtieth Aaai conference on artificial intelligence*, (2016).

[17] Mathias Niepert, 'Discriminative gaifman models', in *Advances in Neural Information Processing Systems*, pp. 3405–3413, (2016).

[18] Matthew Richardson and Pedro Domingos, 'Markov logic networks', *Machine learning*, **62**(1), 107–136, (2006).

[19] Tim Rocktäschel and Sebastian Riedel, 'End-to-end differentiable proving', in *Advances in Neural Information Processing Systems*, pp. 3788–3800, (2017).

[20] Gustav Sourek, Vojtech Aschenbrenner, Filip Zelezny, Steven Schockaert, and Ondrej Kuzelka, 'Lifted relational neural networks: Efficient learning of latent relational structures', *Journal of Artificial Intelligence Research*, **62**, 69–100, (2018).

[21] Charles Sutton and Andrew McCallum, 'Piecewise pseudolikelihood for efficient training of conditional random fields', in *Proceedings of the 24th international conference on Machine learning*, pp. 863–870. ACM, (2007).

[22] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck, 'A semantic loss function for deep learning with symbolic knowledge', in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, (July 2018).

[23] KiJung Yoon, Renjie Liao, Yuwen Xiong, Lisa Zhang, Ethan Fetaya, Raquel Urtasun, Richard Zemel, and Xaq Pitkow, 'Inference in probabilistic graphical models by graph neural networks', *arXiv preprint arXiv:1803.07710*, (2018).