Meta-Interpretive Learning meets Neural Networks

Victor Guimarães[0000−0002−2851−5618] and
Vítor Santos Costa[0000−0002−3344−8237]

CRACS and DCC/FCUP
Universidade do Porto
Porto, Portugal
victorguimaraes13@gmail.com, vsc@dcc.fc.up.pt

Abstract. In this work we present NeuralLog+MIL, a structure learning system based on Meta-Interpretive Learning. NeuralLog+MIL applies the idea of Metagol to learn theories in the NeuralLog language. Metagol is a system that uses a set of higher-order clauses to define a hypotheses space of first-order logic theories and applies those clauses in order to derive first-order theories from examples. Then, NeuralLog transforms a first-order logic program into a neural network. It receives as input a set of first-order clauses that are used to define the neural network models, and a set of facts that becomes weights in the neural network. We show that our approach has competitive results for the link prediction task when compared with Neural-LP in three different datasets. Achieving comparable values for the hit at top 10 entities and the mean rank metric in the UMLS and WordNet datasets; and outperforming Neural-LP in the UWCSE dataset in those metrics.

Keywords: Meta-Interpretive Learning · Neural Networks · Inductive Logic Programming

1 Introduction

NeuralLog is a system developed to transform a first-order logic program into a neural network. It receives as input a set of first-order clauses that are used to define the neural network models, and a set of facts that becomes weights in the neural network. Then, those weights are fine-tuned in order to better describe a set of examples [5].

In this paper, we propose NeuralLog+MIL, a structure learning system that finds first-order logic program for NeuralLog [5]. It is based on the Meta-Interpretive Learning (MIL) system Metagol [12] and uses a higher-order logic theory to define the possible clauses that shall appear in the first-order program. The first-order logic program, then, defines the structure of the neural network, that can then be fine-tuned on the examples.

NeuralLog+MIL contributes to the extensive line of research that is being developed on combining the advantages of high-level knowledge representations, quite often based on logic, and the impressive performance of neural networks. Often, effort relied on pre-existing theories, or on theories learned by different learners, that might not be the best fit for these learners. In fact, previous work on theory learning for Statistical Relational Programs showed progress in this area to be constrained by the need to perform combined inference and by explosive growth in the search space. Thus, most work in this area assumes the existence of an underlying theory. One important exception is Neural-LP [18], a neural network system that is based on TensorLog [2], a system that follows a related approach in order to build neural network models from first-order logic theories.

While neural network has been achieving great performance in a range of tasks [4], it lacks interpretability and explainability. On the other hand, first-order logic struggles to deal with uncertainty and noise; two characteristic inherent to real world problems [11]. Neural-Symbolic Learning and Reasoning is a field of study that tries to combine neural networks with first-order logic [4].

The combination of logic based approaches with neural network can benefit from the strengths of both methods. On one hand, neural networks can bring to first-order logic the capability to deal with uncertainty and noise. On the other hand, first-order logic can improve the performance of neural network models by incorporating background knowledge. This background knowledge might contain expert knowledge that can be used to guide the neural network toward a better model representation. Nevertheless, neural network models based on first-order logic might be easier to interpret and explain, since first-order logic is easier to understand, from a human perspective; and the inference of neural networks based on it is, to some extent, based on the underlying logic structure.

We show that NeuralLog+MIL has comparable results with Neural-LP in the UMLS [8] and the WordNet [10] datasets, for the hit at top 10 and mean rank metrics [1]. Furthermore, NeuralLog+MIL outperforms Neural-LP, in those metrics, in the UWCSE dataset [13].

The remainder of the paper is organized as follows: in Section 2 we give the background knowledge in order to understand this work; in Section 3 we present our structure learning algorithm for NeuralLog using MIL; in Section 4 we present the performed experiments and obtained results; in Section 5 we present the works related to ours; and we conclude and propose directions for future work in Section 6.

2 Background Knowledge

First-order logic programs are basically composed of two parts: (1) a set of logic facts, that represents relations between logic entities; and (2) a set of rules that reason over those facts. In this work, we are concerned about a specific type of rules called a Horn clause [7].

A Horn clause has the form of

$$b(.) \leftarrow p_1(.) \wedge \ldots \wedge p_n(.). \tag{1}$$

where $b(.)$ is called the head of the clause and the set of p_i is called the body and represents a conjunction. The Horn clause states that whenever the body is true, the head must also be true. b and p_i are called *predicates*. In the parentheses, there might be *constants*, that represent logic entities and starts with a lower case letter; or *variables*, which are substituted by a constant in order to prove the rule, and starts with an upper case letter. Constants and variables are generically called *term*. A predicate name, followed by its terms is called an *atom*. Each predicate has a fixed number of terms it accepts, which is called arity, and a predicate p of arity n is represented as p/n. Finally, a *fact* is represented by Horn clauses with empty body and no variables.

There are several algorithms that find the proves of a logic rule. We refer the reader to [14] for a further understanding of logic programming. However, NeuralLog transforms the proof of the rule into the inference of a neural network.

NeuralLog receives as input a set of facts weighted F with arity at most 2 and a set of rules R, both containing a set of entities E. For each distinct predicate $p/2$ of the set of facts, NeuralLog creates a matrix $P \in \mathbb{R}^{|E| \times |E|}$, in such a way that $P_{i,j} = w$, if there is a fact $p(e_i, e_j)$ with weight w in F; otherwise, $P_{i,j} = 0$. Analogously, it represents facts of arity 1 as vectors and facts of arity 0 as scalars. A constant e is represented as an one-hot vector, whose value is 1 for the entry corresponding to the index of e and 0 anywhere else.

Finally, inference of rules are computed by mathematical operations on those matrices, given the numeric representation of the constant for the first term in the head of the rule, and the result is a vector representation of the last term in the head of the rule, in such way that the values different from 0 correspond to logic proved constants. The result values of different rules with the same predicate in the head are summed, in order to obtain the final result for the inference of the predicate.

NeuralLog also allows the definition of numeric functions as predicates, that are applied to the vector representation of the variable of the predicate in the rule. We use this feature to apply activation and output functions to rules.

A neural network is a list of differentiable operations that is applied into an input in order to compute the output. These operations often include internal parameters that are learned by the neural network through a gradient descent algorithm [6]. In NeuralLog, those parameters are represented by the weights of the logic facts.

The concern of this work is to find a suited set of Horn clauses that creates a neural network in order to predict the examples of the task, given the background knowledge, where the examples are pairs of entities that are connected through a specific relation. Given a relation and an (input) entity, we would like to predict all the (output) entities connected to it, through the given relation; this task is called *link prediction*.

NeuralLog creates a neural network for the relation, based on the set of rules, in order to predict the output entities, given the input entity. In addition, the weight of some predicates are specified to be learned by the neural network, through the gradient descent algorithm.

3 Meta-Interpretive Learning with NeuralLog

In this section we introduce NeuralLog+MIL, a structure learning algorithm based on the Meta-Interpretive Learning (MIL) system Metagol [12].

MIL is a method that learns first-order logic theories by the use of a higher-order logic theory that will define the hypotheses space and guide the search of the hypothesis in this space [12].

In first-order logic, the predicate names in the rules, which represent the relations between the logic entities, are constant. In higher-order logic, those predicate names might be variable, and the logic inference system should find the substitution of the name in order to prove the rule.

Metagol uses a modified Prolog meta-interpreter that generates a first-order logic theory that proves the positive examples, without proving the negative ones, given the background knowledge. It does it by traversing a higher-order theory in a similar way a SLD-Resolution algorithm would do [17].

NeuralLog+MIL takes a slightly different approach: instead of adding rules to the first-order theory as needed, it creates a first-order theory by adding all possible rules that satisfies the higher-order program, given a pre-defined depth. Each rule will then receive an associated weight and an activation function.

This approach is better suited to be integrated with NeuralLog, since a single neural network is created and the task to find the weight of the rules is passed to the neural network optimization process. If we had followed Metagol's approach, we would have to create and evaluate several intermediary neural networks, which would be more computationally expensive.

Given a higher-order theory and a target predicate p, NeuralLog+MIL creates a “meta” SLD-Resolution tree, starting with a list of goals $[q]$, replacing its terms by distinct variables, as the root node. Then, for each node in the tree, a child node is created by applying a meta-clause that unifies with one of the goals in the node; in the same way a conventional SLD-Resolution method would. The new child node is created with the list of goals from the parent node, with the goal, whose rule was applied, replaced by the body of the unified meta-clause.

We grow this tree, breadth-first, by applying all possible meta-clauses to all goals in all nodes until we reach the maximum depth. Each edge represents the application of a meta-clause whose head was unified to the used goal in the parent node. For each path from the root to a leaf, we have a meta-program with variable predicates to be instantiated.

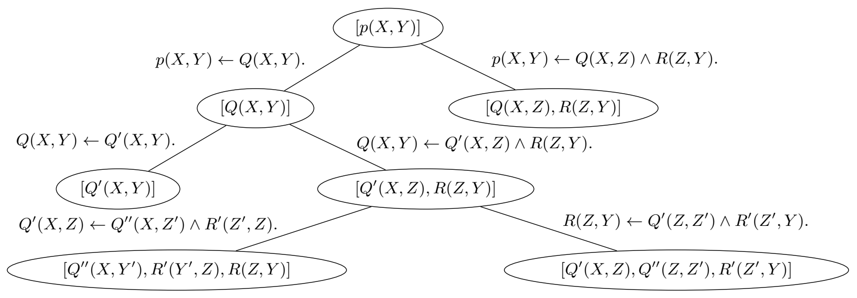
Finally, for each meta-program we generate a first-order program by replacing the set of variable predicates for each possible predicate in the knowledge base. The final program is the concatenation of all generated clauses in all the programs.

For instance, consider the higher-order theory in Table 1 and a target predicate $p/2$. The tree would start with the root node containing the goal $[p(X, Y)]$, referred here as level 0. By applying each clause to the goal of the root node, we would end up with two nodes at level 1: the node $[Q(X, Y)]$, generated by the unified clauses $p(X, Y) \leftarrow Q(X, Y).$; and the node $[Q(X, Z), R(Z, Y)]$, generated by the unified clause $p(X, Y) \leftarrow Q(X, Z) \land R(Z, Y)$. After the addition of both nodes, the level 1 would be complete.

Table 1. Higher-order Logic Theory

(1) $P(X, Y) \leftarrow Q(X, Y).$
(2) $P(X, Y) \leftarrow Q(X, Z) \land R(Z, Y).$

If we would like to go further in the resolution, we could apply the meta-clause to each goal in the nodes and keep adding new nodes until a pre-defined depth. Figure 1 shows an example of part of a tree until depth 3; where some nodes were omitted for clarity.



$[p(X,Y)]$
$p(X,Y) \leftarrow Q(X,Y).$
$p(X,Y) \leftarrow Q(X,Z) \land R(Z,Y).$
$[Q(X,Y)]$
$[Q(X,Z), R(Z,Y)]$
$Q(X,Y) \leftarrow Q'(X,Y).$
$Q(X,Y) \leftarrow Q'(X,Z) \land R(Z,Y).$
$[Q'(X,Y)]$
$[Q'(X,Z), R(Z,Y)]$
$Q'(X,Z) \leftarrow Q''(X,Z') \land R'(Z',Z).$
$R(Z,Y) \leftarrow Q'(Z,Z') \land R'(Z',Y).$
$[Q''(X,Y'), R'(Y',Z), R(Z,Y)]$
$[Q'(X,Z), Q''(Z,Z'), R'(Z',Y)]$

Fig. 1. Meta SLD-Resolution Tree

4 Experiments

In order to demonstrate the capabilities of NeuralLog to learn neural network structure using MIL, we compared NeuralLog+MIL with Neural-LP [18] in three datasets: the Unified Medical Language System (UMLS) [8]; the WordNet dataset [10]; and the UWCSE dataset [13]. We chose to compare NeuralLog+MIL with Neural-LP, since Neural-LP is based on TensorLog, which is closely related to NeuralLog.

4.1 Methodology

We focused at the link prediction task, where we are given a query of the form $q(a, X)$, where the goal is to find all the entities that are related (as second term) to the entity a (as first term) through relation q. For each dataset, we selected a set of target relations and applied the system to learn it. Then, we evaluated the system by using the mean rank, the mean reciprocal rank (MRR) and the hit at top 10 entities, based on the filtered rank of the entities, following the procedure described in [1]. In these experiments, we would like to answer the research question: can NeuralLog+MIL learn the structure representation of NeuralLog models for link prediction tasks?

We use these rank metrics to reproduce the experiments from Neural-LP [18]. Since Neural-LP only uses positive examples, to achieve a fair comparison we give NeuralLog only positive examples as well. Here, we take advantage of the sparse representation of NeuralLog. The positive examples have an associated value of 1, and the missing examples will have an associated value of 0 and the loss function will be able to tune the weights in order to prove only the positive examples. This is equivalent to consider all the missing examples as negatives, which is known as the Close World Assumption (CWA).

UMLS. It is a dataset consisting of 46 relations between biomedical concepts. We selected the most frequent relation (*Affects*) as target relation and applied the system to predict this relation from the remaining ones. We randomly selected 90% of the facts from the target relation as training examples and the other 10% as test. Following the procedure used in [15]. We repeated this process 10 times and reported the average of the runs.

WordNet. It is a dataset consisting of 18 relations between words and it is already split into train, development and test sets. We ran each system once, holding out one relation at a time, and reported the mean results of the relations.

UWCSE. It is already split into 5-folds for the *Advised by* relation. We run the system once for each fold and reported the mean results.

The used meta-theory was the one shown in Table 2, with the depth of 1. It means that each meta-clause is applied directly to the example. For instance, applying the first clause to the example $advisedby(X, Y)$ would generate the clause $advisedby(X, Y) \leftarrow Q(X, Y).$, where Q is replaced for each valid predicate in the knowledge base (KB). If the depth were 2, in addition to replacing Q for each example in the KB, we would also have tried to prove it with another meta-clause; for instance, the last one, which would result in an invented predicate $f(X, Y) \leftarrow Q(X, Z) \wedge R(Z, Y)$.

Table 2. The Meta-Theory Used by NeuralLog+MIL

$$P(X, Y) \leftarrow Q(X, Y).$$
$$P(X, Y) \leftarrow Q(Y, X).$$
$$P(X, Y) \leftarrow Q(X, Z) \wedge R(Z, Y).$$

The head of each generated rule is changed from the target predicate $p/2$ to a predicate $p'/2$ and a rule of the form $p'(X, Y) \leftarrow p(X, Y), output\_function(Y)$.

is added to the theory. Also, each generated rule has two atoms appended to it. The first one is the atom $activation\_function(Y)$, where Y is the output of the rule, and it serves to applying an activation function to the output result of the rule. The second one is an atom of the form $w(id)$, where id is an unique constant for each rule and w is a predicate whose weight will be learned by the network and will multiply the final result of the rule. Finally, we add a rule of the form $p(X, Y) \leftarrow b.$ to the theory. Since this rule does not depend on the input, it will be true for any input and its value will be b, which will be added to the output of the other rules with the predicate $p/2$ in the head, acting as a bias. The value of the predicate b will also be learned by the network.

We used sigmoid as activation function and softmax as output function; and adagrad to minimize the binary cross-entropy with L2 regularization for 50 epochs in the case of the UMLS and UWCSE datasets, and 10 epochs for the WordNet dataset.

We used the default parameters for Neural-LP, which achieved good results in two of the three datasets.

4.2 Results

Table 3 shows the results for each dataset. The lower the mean rank, the better; for the other metrics, the higher, the better. The best value of each metric, for each dataset is bold-faced. Pair of underlined values, of the same metric, in the same dataset, means that the difference between the values has statistical significance according to the two-tailed paired t-test with $p < 0.05$.

Table 3. Results for the Filtered Rank Metric

Dataset	NeuralLog+MIL			Neural-LP		
	Hit @ 10	Mean Rank	MRR	Hit @ 10	Mean Rank	MRR
UMLS	0.9894	1.6056	0.8856	**0.9970**	**1.2773**	**0.9303**
WordNet	0.8216	10.0647	0.6221	**0.9418**	**9.2479**	**0.9298**
UWCSE	**0.9165**	**5.5818**	0.2158	0.4446	36.8060	**0.2458**

As can be seen from the table, NeuralLog+MIL shows a competitive result with Neural-LP in both the UMLS and WordNet datasets; and is considerably better in the UWCSE dataset, for the hit at top 10 entities and the mean rank.

WordNet is composed of 18 relations with a different number of examples for each relation. If we take the average of the metrics, weighted by the number of examples of each relation, the difference between NeuralLog+MIL and Neural-LP is even smaller, being 0.9280 against 0.9544, for the hit at top 10 entities; and 4.1480 against 6.1228, for the mean rank; for the NeuralLog+MIL and Neural-LP, respectively. In this scenario, the mean rank of NeuralLog+MIL is better than the one of Neural-LP.

Finally, from these results, we show that NeuralLog+MIL can learn the structure of NeuralLog models for link prediction tasks, affirmatively stating that

NeuralLog+MIL can learn structure representations of NeuralLog models for link prediction tasks.

5 Related Work

A strong feature of NeuralLog is that it can be used together with any other neural network structure. In order to achieve this, the remaining of the neural network must be defined in the logic theory. TensorLog [2] is also capable of such a thing. However, Neural-LP [18], which uses the TensorLog inference mechanism, does not accept a pre-existing theory and finds all the rules by itself. In this way, it is only suited to pure relational logic tasks.

DeepProbLog also combines neural networks with first-order logic. However, its semantic is different from NeuralLog [9]. DeepProbLog is based on ProbLog, which uses the semantic of possible worlds [3]. It allows the definition of neural network predicates, which are predicates whose underlying implementation is a neural network. The optimization and inference of the neural network is combined with the world semantic of the ProbLog system. On the other hand, NeuralLog creates a neural network to simulate logic inference through differentiable operations.

Traditional Inductive Logic Programming (ILP) systems try to find logic theories to prove a set of examples, given background knowledge in a pure logic fashion [11]. NeuralLog+MIL extends this idea to find the neural network structure of NeuralLog by using MIL [12]. MIL is well suited to integrate with NeuralLog, since the higher-order theory allows the user to define a template in order to create the relational part of the logic theory. This template can be used to append the relational part to an existing theory, which might include the definition of an existing neural network.

Furthermore, the higher-order theory may also be used to find the logic part that integrates with the neural network part, by specifying a constant predicate that will pose as a connection point between the logic part and the neural network. However, we left this for a future work.

The idea of applying MIL to systems based on first-order logic is not novel. A system that does that is the Iterated Structural Gradient (ISG) [15], which is based on ProPPR [16], an SLP system. However, ProPPR uses a different inference mechanism that cannot be easily integrated with deep learning.

6 Conclusion

In this paper we presented NeuralLog+MIL, a structure learning system for the NeuralLog language [5], based on MIL [12]. We compared our approach with Neural-LP, a system that learns structure for TensorLog [2].

We showed that our approach has competitive results in the rank metrics for the UMLS [8] and WordNet [10] datasets. In addition, it outperforms Neural-LP in the UWCSE dataset [13], according to the hit at top 10 and the mean rank metrics. It is important to notice that we trained our system using reasonable parameters and our goal was not to find the best possible model for each dataset, thus, we believe the results can still be improved by the search of better parameters.

Furthermore, NeuralLog+MIL is suited to learn logic theories alongside other neural network definitions in the NeuralLog language, while Neural-LP can only learn the entire logic theory from scratch. However, we left the analysis of models that combine logic parts with other neural network structures for future works.

We would also like to apply NeuralLog+MIL in relational tasks where (partially correct) theories are available in order to analyse if NeuralLog+MIL is able to improve the quality of the existing theory by appending clauses built by the higher-order theory.

In addition, we would also like to experiment learning with a deeper meta-theory in future works, which could take advantage of MIL capability of inventing new predicates [12].

Acknowledgment Victor Guimarães was financed by the Portuguese funding agency, FCT — Fundação para a Ciência e a Tecnologia through Ph.D. scholarships 2020.05718.BD.

References

1. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Burges, C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 26, pp. 2787–2795. Curran Associates, Inc. (2013)
2. Cohen, W.W., Yang, F., Mazaitis, K.: Tensorlog: A probabilistic database implemented using deep-learning infrastructure. J. Artif. Intell. Res. **67**, 285–325 (2020). https://doi.org/10.1613/jair.1.11944
3. De Raedt, L., Kimmig, A., Toivonen, H.: ProbLog: A probabilistic prolog and its application in link discovery. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence. pp. 2468–2473. IJCAI'07, Morgan Kaufmann Publishers Inc., Hyderabad, India (January 2007)
4. Garcez, A.d., Besold, T.R., De Raedt, L., Földiak, P., Hitzler, P., Icard, T., Kühnberger, K.U., Lamb, L.C., Miikkulainen, R., Silver, D.L.: Neural-symbolic learning and reasoning: contributions and challenges. In: 2015 AAAI Spring Symposium Series (2015)
5. Guimarães, V., Costa, V.S.: Neurallog: A neural logic language (2021)
6. Haykin, S.: Neural networks, vol. 2. Prentice hall New York (1994)
7. Horn, A.: On sentences which are true of direct unions of algebras. The Journal of Symbolic Logic **16**(1), 14–21 (1951)
8. Kok, S., Domingos, P.: Statistical predicate invention. In: Proceedings of the 24th International Conference on Machine Learning. p. 433–440. ICML '07, Association for Computing Machinery, New York, NY, USA (2007)
9. Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., De Raedt, L.: DeepProbLog: Neural probabilistic logic programming. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in

Neural Information Processing Systems 31, pp. 3749–3759. Curran Associates, Inc. (2018)
10. Miller, G.A.: Wordnet: A lexical database for english. Commun. ACM **38**(11), 39–41 (Nov 1995)
11. Muggleton, S., De Raedt, L.: Inductive logic programming: Theory and methods. J. Log. Program. **19/20**, 629–679 (1994)
12. Muggleton, S.H., Lin, D., Tamaddoni-Nezhad, A.: Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. Mach. Learn. **100**(1), 49–73 (2015)
13. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning **62**(1), 107–136 (2006)
14. Shan-Hwei Nienhuys-Cheng, R.d.W.a.: Foundations of Inductive Logic Programming. Lecture Notes in Computer Science 1228 : Lecture Notes in Artificial Intelligence, Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 1 edn. (1997)
15. Wang, W.Y., Mazaitis, K., Cohen, W.W.: Structure learning via parameter learning. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. p. 1199–1208. CIKM '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2661829.2662022
16. Wang, W.Y., Mazaitis, K., Lao, N., Cohen, W.W.: Efficient inference and learning in a large knowledge base. Machine Learning **100**(1), 1–26 (2015)
17. Warren, D.H.D., Pereira, L.M., Pereira, F.: Prolog - the language and its implementation compared with lisp. SIGPLAN Not. **12**(8), 109–115 (1977)
18. Yang, F., Yang, Z., Cohen, W.W.: Differentiable learning of logical rules for knowledge base reasoning. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 2319–2328. Curran Associates, Inc. (2017)