Lifted Relational Neural Networks

Gustav Šourek
Czech Technical University
Prague, Czech Republic
souregus@fel.cvut.cz

Vojtěch Aschenbrenner
Charles University
Prague, Czech Republic
v@asch.cz

Filip Železný
Czech Technical University
Prague, Czech Republic
zelezny@fel.cvut.cz

Ondřej Kuželka*
Cardiff University
Cardiff, United Kingdom
KuzelkaO@cardiff.ac.uk

Abstract

We propose a method combining relational-logic representations with neural network learning. A general lifted architecture, possibly reflecting some background domain knowledge, is described through relational rules which may be handcrafted or learned. The relational rule-set serves as a template for unfolding possibly deep neural networks whose structures also reflect the structures of given training or testing relational examples. Different networks corresponding to different examples share their weights, which co-evolve during training by stochastic gradient descend algorithm. Discovery of notable latent relational concepts and experiments on 78 relational learning benchmarks demonstrate favorable performance of the method.

1 Introduction

Lifted models also known as *templated* models have attracted significant attention recently [10] in areas such as statistical relational learning. Lifted models define patterns from which specific (ground) models can be unfolded. For example, a lifted Markov network model [18] may express that *friends of smokers tend to be smokers* and such a pattern then constrains the probabilistic relationships in all sets of vertices corresponding to particular friends-smokers in the derived ground Markov network. The lifted patterns are typically encoded in relational logic-based languages.

Here we contribute a method for (deep) lifted feed-forward neural network learning, in which the ground network structure is unfolded from a set of weighted rules in relational logic. The relational rules are instantly interpretable and can be handcrafted by a domain expert or learned, e.g. through techniques of Inductive Logic Programming (ILP) [5]. Weights of the ground neural networks are determined by the weighted relational rules and can be learned by stochastic gradient descend algorithm. This means that weights between different ground neurons constructed from the same relational rule are tied in our framework, similarly to how weights are shared in lifted graphical models in statistical relational learning or how weights are tied together by application of filters in convolutional neural networks in deep learning. A salient property of our approach distinguishing it from previous studies on adapting neural networks for relational learning is that the ground network structure depends not only on the relational rule set but also on a particular example, i.e., different networks are constructed for different examples to exploit their particular relational properties. However, the different networks share their weights as these are all bound to the relational rules, and so weight-updates performed for one training example are reflected in networks produced for other examples, which allows the model to learn directly from relational data.

*Corresponding author.

The main advantage of the presented approach is that it can effectively learn weights of latent non-ground relational structures, which is close in principle to predicate invention in ILP [5]. This is a difficult task for existing lifted systems based on probabilistic inference because there one typically needs to run expensive expectation maximization algorithms in order to learn parameters when latent structures are present. On the other hand, deep neural networks, which we exploit in our work, have been shown to effectively learn latent structures, although obviously only in the ground non-relational settings. By combining relational logic with deep neural networks, we obtain a framework flexible enough to learn weights of latent relational structures, which we also verify experimentally. While there have been several works combining propositional or relational logic with neural networks [21, 3, 6], none of the existing methods is able to learn weights of latent non-ground relational structures [1].

2 Lifted Relational Neural Networks

A lifted relational neural network (LRNN) $\mathcal{N}$ is a set of weighted definite clauses, i.e. pairs (R_i, w_i) where R_i is a function-free definite clause and w_i is a real number. When $\mathcal{N}$ is a set of weighted definite clauses, $\mathcal{N}^*$ will denote the corresponding set of the definite clauses without weights, i.e. $\mathcal{N}^* = \{C : (C, w) \in \mathcal{N}\}$. The set $\mathcal{N}$ must satisfy the following *non-recursiveness*[2] requirement: there must exist a strict ordering $\prec$ of predicates such that if there is a rule with a predicate p_1 in the head and a predicate p_2 in the body then $p_1 \prec p_2$.

Given a LRNN $\mathcal{N}$, let $\mathcal{H}$ be the least Herbrand model of $\mathcal{N}^*$. We define *grounding of the LRNN* $\mathcal{N}$ as $\overline{\mathcal{N}} = \{(h\theta \leftarrow b_1\theta \wedge \cdots \wedge b_k\theta, w) : (h \leftarrow b_1 \wedge \cdots \wedge b_k, w) \in \mathcal{N}$ and $\{h\theta, b_1\theta, \ldots, b_k\theta\} \subseteq \mathcal{H}\}$. That is, $\overline{\mathcal{N}}$ is defined as the set of ground definite clauses which can be obtained by grounding rules from the LRNN and which are active in the least Herbrand model of $\mathcal{N}^*$ (a rule is active in $\mathcal{H}$ if its body is true in $\mathcal{H}$). As already outlined in Introduction, LRNNs are templates for creating *ground* neural networks. The requirement that ground rules should be active in $\mathcal{H}$ is beneficial for practice because it provides us with flexibility in controlling complexity and tractability of the constructed ground neural networks.

Example 1. Let

$$\begin{aligned}\mathcal{N} =\{&(\mathit{mother}(C, M) \leftarrow \mathit{parent}(C, M) \wedge \mathit{female}(M), 1),\\ &(\mathit{father}(C, F) \leftarrow \mathit{parent}(C, F) \wedge \mathit{male}(F), 2),\\ &(\mathit{female}(\mathit{alice}), 1), (\mathit{parent}(\mathit{bob}, \mathit{alice}), 1), (parent(\mathit{eve}, \mathit{alice}), 1)\}.\end{aligned}$$

Then for its grounding we have

$$\begin{aligned}\overline{\mathcal{N}} =\{&(\mathit{mother}(\mathit{bob}, \mathit{alice}) \leftarrow \mathit{parent}(\mathit{bob}, \mathit{alice}) \wedge \mathit{female}(\mathit{alice}), 1),\\ &(\mathit{mother}(\mathit{eve}, \mathit{alice}) \leftarrow \mathit{parent}(\mathit{eve}, \mathit{alice}) \wedge \mathit{female}(\mathit{alice}), 1),\\ &(\mathit{female}(\mathit{alice}), 1), (\mathit{parent}(\mathit{bob}, \mathit{alice}), 1), (\mathit{parent}(\mathit{eve}, \mathit{alice}), 1)\}.\end{aligned}$$

Notice that $\overline{\mathcal{N}}$ does not contain the predicates $\mathit{male}/1$ or $\mathit{father}/2$ as there are no ground atoms based on them in the least Herbrand model of $\mathcal{N}$.

Definition 1. *Let $\mathcal{N}$ be a LRNN, and let $\overline{\mathcal{N}}$ be its grounding. Let $g_\vee$, $g_\wedge$ and $g^*_\wedge$ be families of multivariate functions with exactly one function for each number of arguments. The* ground neural network *of $\mathcal{N}$ is a feedforward neural network constructed as follows.*

- *For every ground atom h occurring in $\overline{\mathcal{N}}$, there is a neuron A_h, called* atom neuron*. The activation functions of atom neurons are from the family $g_\vee$.*
- *For every ground fact $(h, w) \in \overline{\mathcal{N}}$, there is a neuron $F_{(h,w)}$, called* fact neuron*, which has no input and always outputs a constant value.*

[1] Exemplification of latent non-ground relational concept learning and a more detailed description of Lifted Relational Neural Networks can be found in [20].

[2] The reason why we do not allow recursion will be clearer when we explain weight learning in the next section. Here, we just note that rule sets without recursion will allow us to directly exploit gradient descent training of feed-forward neural networks.

- *For every ground rule* $h\theta \leftarrow b_1\theta \wedge \cdots \wedge b_k\theta \in \overline{\mathcal{N}}^*$*, there is a neuron* $R_{h\theta \leftarrow b_1\theta \wedge \cdots \wedge b_k\theta}$*, called* rule neuron*. It has the atom neurons* $A_{b_1\theta}, \ldots, A_{b_k\theta}$ *as inputs, all with weight* 1*. The activation functions of rule neurons are from the family* $g_\wedge$*.*
- *For every rule* $(h \leftarrow b_1 \wedge \cdots \wedge b_k, w) \in \mathcal{N}$ *and every* $h\theta \in \mathcal{H}$*, there is a neuron* $Agg^{h\theta}_{(h \leftarrow b_1 \wedge \cdots \wedge b_k, w)}$*, called* aggregation neuron*. Its inputs are all rule neurons* $R_{h\theta' \leftarrow b_1\theta' \wedge \cdots \wedge b_k\theta'}$ *where* $h\theta = h\theta'$ *with all weights equal to* 1*. The activation functions of the aggregation neurons are from the family* $g^*_\wedge$*.*
- *Inputs of an atom neuron* $A_{h\theta}$ *are the aggregation neurons* $Agg^{h\theta}_{(h \leftarrow b_1 \wedge \cdots \wedge b_k, w)}$ *and fact neurons* $F_{(h\theta, w)}$*. The weights of the input neurons are the respective* w*'s.*

Example 2. Let us consider the following LRNN

$$\mathcal{N} = \{(\mathit{foal}(A) \leftarrow \mathit{parent}(A,P) \wedge \mathit{horse}(P), w_m), (\mathit{foal}(A) \leftarrow \mathit{sibling}(A,S) \wedge \mathit{horse}(S), w_n),$$
$$(\mathit{horse}(\mathit{dakotta}), w_1), (\mathit{horse}(\mathit{cheyenne}), w_2), (\mathit{horse}(\mathit{aida}), w_3),$$
$$(\mathit{parent}(\mathit{star}, \mathit{aida}), w_6), (\mathit{parent}(\mathit{star}, \mathit{cheyenne}), w_5), (\mathit{sibling}(\mathit{star}, \mathit{dakotta}), w_4)\}.$$

The LRNN $\mathcal{N}$ and its ground neural network are shown in Fig. 1.

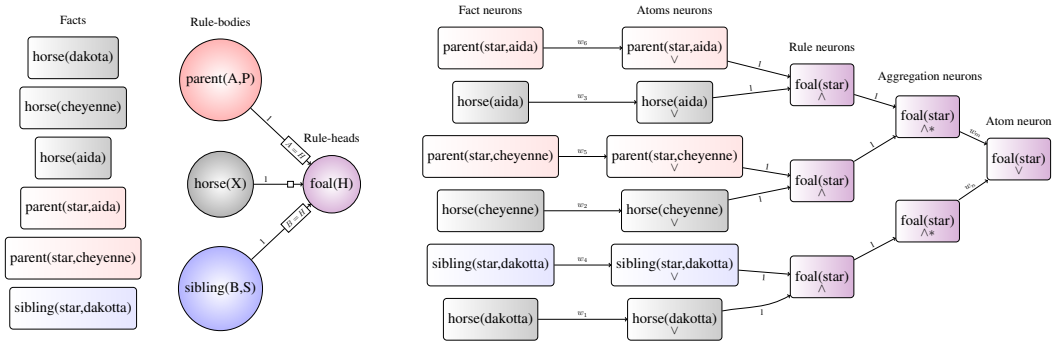

Figure 1: Depiction of the rule-based template (left) of LRNN $\mathcal{N}$ from Ex. 2, and its corresponding ground neural network $\overline{\mathcal{N}}$ (right), with colors denoting the predicate signatures, rectangular nodes corresponding to ground and circular to lifted literals, respectively.

What distinguishes LRNNs from ordinary neural networks the most is the following property. Having a pre-trained LRNN $\mathcal{N}$ described by some general rules, we can extend it with description of a particular case to obtain a ground neural network and then use the latter for prediction. This is similar in spirit to lifted graphical models.

Example 3. For instance, $\mathcal{N}$ may describe general rules for explosiveness of molecules (e.g., represented by a predicate *explosive*) and $\mathcal{M}_1$ and $\mathcal{M}_2$ may be sets of (weighted) facts describing two particular molecules. Then to use the LRNN $\mathcal{N}$ for predicting whether $\mathcal{M}_1$ and $\mathcal{M}_2$ are explosive, we can simply construct ground NNs of $\overline{\mathcal{N} \cup \mathcal{M}_1}$ and $\overline{\mathcal{N} \cup \mathcal{M}_2}$, and compute the output of the respective atom neurons $\mathit{explosive}^1 \in \overline{\mathcal{N} \cup \mathcal{M}_1}$ and $\mathit{explosive}^2 \in \overline{\mathcal{N} \cup \mathcal{M}_2}$. As a distinctive feature of lifted models, the two ground LRNNs for the two example molecules may have very different size and structure because the least Herbrand models of $\mathcal{N}^* \cup \mathcal{M}_1^*$ and of $\mathcal{N}^* \cup \mathcal{M}_2^*$, which determine the structures of the ground LRNNs, may be very different (because the structure and the size of the molecules described by $\mathcal{M}_1$ and $\mathcal{M}_2$ are different). An illustration of this effect, for two example molecules and a template $\mathcal{N}$ from Fig. 2, is displayed in Fig. 3.

Depending on the used families of activation functions $g_\vee$, $g_\wedge$ and $g^*_\wedge$, we can obtain neural networks with different behavior. For intuitiveness, in order for rules $(h \leftarrow b_1 \wedge \cdots \wedge b_k, w)$ to behave similarly to "if-then" rules, we should prefer the outputs of rule neurons to be *high* (e.g. close to 1) if and only if all the inputs from the atom neurons corresponding to the literals from the body of the rule have high outputs. Similarly, we should prefer the output of the atom neurons, which should intuitively behave similarly to disjunction, to be high if and only if at least one of the rule neurons or fact neurons, which are inputs for the given atom neuron, has high output. Logical operators from various fuzzy logics [11] may serve as an inspiration for selecting suitable activation functions.

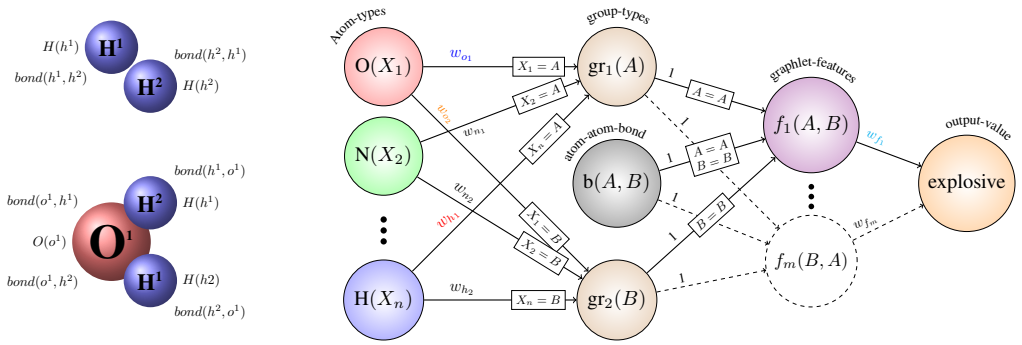

Figure 2: Two example molecules (left), described by surrounding sets of ground facts $\mathcal{M}_1$ and $\mathcal{M}_2$, are being merged with the lifted LRNN $\mathcal{N}$, composed of general weighted rules loosely pointing to explosiveness of molecules (right), to form two ground networks displayed in Fig. 3. The rules in $\mathcal{N}$ provide adaptive means to create latent groups (gr_i) of atom types ($O \dots H$) that, through a bond predicate ($b(A, B)$) connecting couples of atoms, form relational features (e.g., $f_1(A, B) \leftarrow gr_1(A) \wedge bond(A, B) \wedge gr_2(B)$), which set the basis for the final explosiveness output. For the sake of space we assume a single relational (graphlet) feature f_1 only.

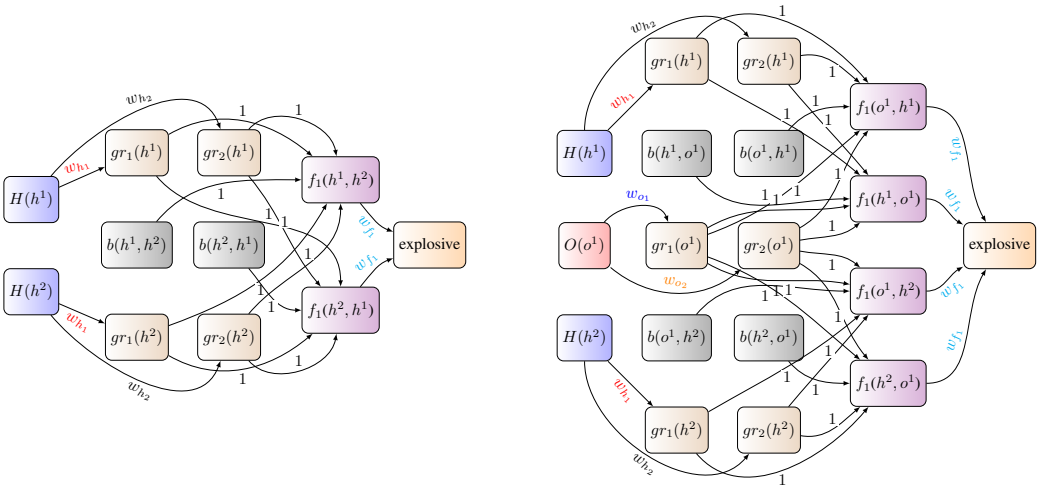

Figure 3: Two groundings $\overline{\mathcal{N} \cup \mathcal{M}_1}$ and $\overline{\mathcal{N} \cup \mathcal{M}_2}$ formed by merging the two example molecules with the LRNN $\mathcal{N}$ from Fig. 2. The shared predicate signatures and weights tied by the template are denoted by colors. For the sake of space we display only ground rule sets instead of complete ground networks (i.e., fact and aggregation neurons are omitted), Fig. 1 illustrates the (direct) correspondence of such a set to a full ground neural network.

Example 4. In Goedel fuzzy logic, conjunction $b_1 \wedge \dots \wedge b_k$, where b_i are fuzzy logic literals, is given as $\min_i b_i$ and disjunction $b_1 \vee \dots \vee b_k$ is given as $\max_i b_i$. To emulate reasoning in Goedel logic, we could simply set $g_\wedge(b_1, \dots, b_k) = \min_i b_i$, $g^*_\wedge(b_1, \dots, b_m) = \max_i b_i$, and $g_\vee(b_1, \dots, b_m) = \max_i b_i$. Here, the output of any rule neuron $R_{h \leftarrow b_1 \wedge \dots \wedge b_k}$ is the minimum value which makes the fuzzy truth value of the implication $h \leftarrow b_1 \wedge \dots \wedge b_k$ equal to 1 in the Goedel fuzzy logic. Likewise, the output of any aggregation neuron is the minimum value which makes the fuzzy truth value of all the respective ground implications equal to 1 simultaneously. This way, LRNNs can emulate fuzzy logic programming.

Next, we introduce two particular collections of activation functions inspired by fuzzy logic which will be used in the experiments (note that the activation functions shown in the above example would not be very suitable for gradient-based learning).

Definition 2 (Max-Sigmoid Activation Functions)**.** *The Max-Sigmoid (MS) collection of activation functions is composed of the following three families of functions:* $g_\wedge(b_1,\ldots,b_k) = \mathit{sigm}\left(\sum_{i=1}^{k} b_i - k + b_0\right)$, $g^*_\wedge(b_1,\ldots,b_m) = \max_i b_i$, *and* $g_\vee(b_1,\ldots,b_k) = \mathit{sigm}\left(\sum_{i=1}^{k} b_i + b_0\right)$.

The rationale for this family of activation functions is as follows. As already mentioned, the activation function $g_\wedge$ should have high output if and only if all its inputs are high. To achieve this, we can crudely approximate Lukasiewicz fuzzy conjunction, which is given as $\max\{0, b_1+\cdots+b_k-k+1\}$, by the function $\mathit{sigm}\,(b_1 + \cdots + b_k - k + b_0)$. The activation function $g^*_\wedge$ outputs the value equal to the highest of its inputs. Example 5 illustrates that this can be seen as finding the best "match" of a pattern (rule). The activation function $g_\vee$ should have high output if at least one of the inputs is high or if all inputs are somewhat high. To satisfy this, we can crudely approximate Lukasiewicz fuzzy disjunction, which is given as $\min\{1, b_1+\cdots+b_k\}$ by the function $\mathit{sigm}\,(b_1 + \cdots + b_k + b_0)$. Example 6 illustrates the intuition for the activation function $g_\vee$.

Example 5. Let us consider the LRNN

$$\begin{aligned}\mathcal{N} =\{&(\mathit{hasBrightEdge} \leftarrow \mathit{isBright}(E), 1), (\mathit{isBright}(E) \leftarrow \mathit{edge}(E,U,V) \wedge \mathit{bright}(U) \wedge \mathit{bright}(V),\\ &1), (\mathit{bright}(U) \leftarrow \mathit{yellow}(U), 2), (\mathit{bright}(U) \leftarrow \mathit{red}(U), 1), (\mathit{bright}(U) \leftarrow \mathit{blue}(U), 0.5)\}.\end{aligned}$$

Let us also have a set $\mathcal{G}$ describing a graph with colored vertices.

$$\begin{aligned}\mathcal{G} =\{&(\mathit{edge}(e_1,v_1,v_2),1), (\mathit{edge}(e_2,v_2,v_3),1), (\mathit{edge}(e_3,v_3,v_4),1), (\mathit{edge}(e_4,v_4,v_1),1),\\ &(\mathit{red}(v_1),1), (\mathit{blue}(v_2),1), (\mathit{yellow}(v_3),1), (\mathit{yellow}(v_4),1)\}\end{aligned}$$

The output of the atom neuron $A_{\mathit{hasBrightEdge}}$ will only depend on the "brightest edge", i.e. in this case on the edge e_3. The output would be the same for any other colored graph $\mathcal{G}'$, which would also contain an edge connecting two yellow vertices. Thus, for instance, if we considered some physicochemical property of atoms (e.g. their partial charge) instead of brightness of colors, and molecules instead of colored graphs, the corresponding networks could detect presence of a molecular substructure similar to a prescribed pattern.

Example 6. Let us have the LRNN

$$\begin{aligned}\mathcal{N} =\{&(\mathit{highPressure}(X) \leftarrow \mathit{stressed}(X), 1), (\mathit{highPressure}(X) \leftarrow \mathit{obese}(X), 1),\\ &(\mathit{highPressure}(X) \leftarrow \mathit{exercises}(X), -1)\}\end{aligned}$$

and the set of weighted facts $\mathcal{P} = \{(\mathit{stressed}(\mathit{alice}), 1),\ (\mathit{obese}(\mathit{alice}), 1),\ (\mathit{stressed}(\mathit{bob}), 1),\ (\mathit{exercises}(\mathit{bob}), 1)\}$. Outputs of aggregation neurons corresponding to rules from $\mathcal{N}$ with the same predicate in the head are combined using the activation functions $g_\vee$. Intuitively, rules and facts with the same predicate in the head can be seen as forming a logistic regression on the values given by the aggregation neurons from the lower layers. When the LRNN has just one layer, as in this example, one can achieve the same effect using techniques from *propositionalization* [12] – treating the bodies of the rules as features and feeding them as attributes to a logistic regression classifier. However, as soon as the LRNN has more layers, this effect cannot be emulated using propositionalization. In this particular example, if we construct the ground LRNN of $\overline{\mathcal{N} \cup \mathcal{P}}$ then the output of the atom neuron $A_{\mathit{highPressure}(\mathit{alice})}$ will be higher than the output of the atom neuron $A_{\mathit{highPressure}(\mathit{bob})}$ (because *alice* is stressed and obese whereas *bob* is just stressed and exercises).

The Max-Sigmoid activation function is obviously not the only one possible. It is useful when we are interested in detecting one or more patterns (such as the existence of an edge as bright as possible in Example 5) but less useful in situations similar to the one depicted in the next example.

Example 7. Let us consider the following simple LRNN for predicting individuals infected by flu

$$\mathcal{N} =\{(\mathit{hasFlu}(A) \leftarrow \mathit{friends}(A,B) \wedge \mathit{hasFluDiagnosed}(B), 1)\}$$

and a set of weighted ground facts $\mathcal{P}$ about a group of people and their friendships. If we constructed the ground neural networks of $\overline{\mathcal{N} \cup \mathcal{P}}$ using the activation functions from the Max-Sigmoid family then the prediction of whether an individual has flu would be entirely based on the existence of at least one person who already had flu diagnosed. It would be obviously more meaningful to base the predictions on the fraction of one's friends who had flu diagnosed.

A family of activation functions which are more appropriate in situations similar to to the one described in the above example is given by the next definition.

Definition 3 (Avg-Sigmoid Activation Functions)**.** *The Avg-Sigmoid (AS) collection of activation functions is composed of the following three families of functions:* $g_\wedge(b_1,\ldots,b_k) = sigm\left(\sum_{i=1}^{k} b_i - k + b_0\right)$, $g^*_\wedge(b_1,\ldots,b_m) = \frac{1}{m}\sum_{i=1}^{m} b_i$, *and* $g_\vee(b_1,\ldots,b_k) = \sum_{i=1}^{k} b_i + b_0$.

Another advantage of the Avg-Sigmoid family of activation functions over the Max-Sigmoid family is also that the functions from the Avg-Sigmoid family are everywhere differentiable (which simplifies learning). We note that other activation function families based on combinations of different aggregation functions might also be exploited for LRNN learning. Further learning scenarios and LRNN modeling constructs can be found in [20].

3 Weight Learning

Let us have a LRNN $\mathcal{N}$ and a set of training examples $\mathcal{E} = \{\mathcal{E}^1,\ldots,\mathcal{E}^m\}$ where each $\mathcal{E}^j$ is some structure represented by a set of weighted propositions (e.g., left part of Fig. 2), i.e. a LRNN containing only facts[3]. Let us also have a set $\mathcal{Q} = \{\{(q^1_1,t^1_1),\ldots,(q^1_{k_1},t^1_{k_1})\},\ldots,\{(q^m_1,t^m_1),\ldots,(q^m_{k_m},t^m_{k_m})\}\}$ where q^j_i are ground atoms, which we call *training query atoms*, and t^j_i are their *target values*. For any query atom q^j_i, let y^j_i denote the output of the atom neuron $A_{q^j_i}$ in the ground neural network of $\overline{\mathcal{N}\cup\mathcal{E}^j}$. The goal of the learning process is to find weights w_h of the rules (and possibly facts) in $\mathcal{N}$ minimizing cost J on the training query atoms $J(\mathcal{Q}) = \sum_{j=1}^{m}\sum_{i=1}^{k_j} cost(y^j_i,t^j_i)$ where *cost* is some predefined cost function which measures the discrepancy between the output of the atom neurons of the training query atoms and their desired target values. Similarly to conventional NNs, weight adaptation is performed by gradient descent steps $w_h \leftarrow w_h - \gamma\frac{\partial J(\mathcal{Q})}{\partial w_h}$ where γ is some given learning rate. The main difference is that in the case of LRNNs, the ground neural networks may be very different for different learning examples $\mathcal{E}^j$. However, this is not a fundamental problem because the weights for all the ground neural networks $\overline{\mathcal{N}\cup\mathcal{E}^j}$ are fully specified in the LRNN $\mathcal{N}$. Moreover, the weights from $\mathcal{N}$ can be repeated multiple times within a single $\overline{\mathcal{N}\cup\mathcal{E}^j}$, but since recursion is not allowed, the same weight can appear at most once on any simple path from a fact neuron to an atom neuron. Therefore it is possible to learn the weights using conventional online stochastic gradient descent algorithm[4], except that the increments for the shared weights must be accumulated, which is a simple consequence of linearity of partial differentiation.

Specifically, our weight-learning algorithm works as follows. First, it grounds the given LRNN $\mathcal{N}$ w.r.t. every example $\mathcal{E}^j$ from the dataset which gives it a set of ground neural networks $\overline{\mathcal{N}\cup\mathcal{E}^j}$ with shared weights (it keeps the information about the origin of each weight so that it could update the respective weights in the template in each step of the iteration). It then iterates over the ground networks in a random order, computes gradient of the error function for the current particular example given the current weights in the template, updates the weights accordingly and continues iterating these steps (i.e., the standard stochastic gradient descent procedure). In order to reduce the risk of getting stuck in poor quality local optima, we also employ a restart strategy for this algorithm. A more detailed version of LRNN weight learning can be found in [20].

4 Related Work

The main inspiration for the work presented in this paper are lifted graphical models such as Markov logic networks [18] or Bayesian logic programs [9]. However, none of these existing lifted graphical models is particularly well suited for learning parameters of latent relational structures. Our approach is also generally related to prior art in combining logical rules with neural networks, also known as neural-symbolic integration [4], such as in the KBANN system. While the KBANN [21] also constructs the network structure from given rules, these rules are propositional rather than relational and do not serve as a lifted template. Therefore it is impossible to learn relational latent

[3]The restriction of learning from facts only is actually not necessary but it will simplify this presentation.

[4]Learning is slightly more complicated for LRNNs with the Max-Sigmoid family of activation functions because the max operator introduces non-differentiable points to the optimization problem.

structures such as soft clustering of first-order-logic constants. A more recent system CILP++[6] utilizes a relational representation, which is however converted into a propositional form through a propositionalization technique [12]. This again means that latent non-ground relational structures cannot be learned by CILP++ either. A somewhat more closely related paper of FONN method [3] also designs a technique forming a network from relational rule set however this rule set is flat, producing only 1-layer (shallow) networks in which relational patterns are not hierarchically aggregated. While there are many other approaches of neural-symbolic integration aiming at relational (and first-order) representations [1], e.g. based on the CORE method [8], they typically search for a uniform model of the logic program in scope and thus principally differ from the presented lifted modeling approach.

While standard feed-forward neural networks can be seen as a special case of LRNNs, since any such a fixed neural architecture can be encoded in a corresponding ground rule set with respective activation functions, a salient aspect of our method is that it allows for learning from structured (relational) examples, rather than just attribute vectors. There has been previous work on adapting neural networks to cope with certain facets of relational representations. For example, extension to multi-instance learning was presented in [17]. A similarly directed work [2] facilitated aggregative reasoning to process sets of related tuples from relational database as a sequence through recurrent neural network structure, which was also presented for more general structures in [19]. These approaches are principally different from the presented method as they do not follow the lifted modeling strategy to cope with variations in structure of relational samples.

5 Experiments

In this section we describe experiments performed on 78 datasets of organic molecules: Mutagenesis dataset [15], four datasets from the predictive toxicollogy challenge and 73 NCI-GI datasets [16]. The Mutagenesis dataset contains 188 molecules with labels denoting their mutagenicity. A number of the results published on the mutagenesis dataset use extended set of features, providing additional expert knowledge on properties of molecules, degrading the role of learning capabilities in relational models (i.e. the expert features are discriminative enough by themselves). We do not use any of the extra features as we utilize only atom-bond information. The predictive toxicology challenge dataset (PTC) [7] is composed of four datasets of molecules labeled by their toxicity for female rats (fr), mouse (fm) and male rat (mr) and mouse (mm). Each of the NCI-GI datasets contains several thousands of molecules labeled by their ability to inhibit growth of different types of tumors. We compare performance of LRNNs to state-of-the-art relational learners kFOIL [13] and nFOIL [14], where kFOIL combines relational rule learninng with support vector machines and nFOIL combines relational rule learning with naive Bayes learning.

For LRNNs we use a simple hand-crafted template which is principally identical to the template discussed in Figure 2. Using such a generic template for all the datasets, we make sure that there is no additional expert knowledge involved [5]. The idea is that in the process of learning, useful latent relational concepts are created within the neural network by the means of weight adaptation rather than by explicit enumeration, in contrast to propositional approaches and ILP [5]. Indeed, none of the rules used in this template is useful on itself for prediction as a hard logic rule without weight adaptation.

To set the parameters of LRNNs we use the empirical risk minimization principle on the training cross-validation folds to select the parameters such as step size, restarts, number of iterations, etc. This way we obtain unbiased estimates of performance of our methods since test data is never involved in parameter selection. The time for training a LRNN was in the order of few hours for the larger NCI-GI datasets. The results of the experiments are summarized in Figure 4. LRNNs perform clearly the best of the algorithms in terms of accuracy as they have lower prediction error than kFOIL and nFOIL on significant majority of datasets. We also tried to compare LRNNs with another recent algorithm combining logic and neural networks called CILP++ [6], but we were not able to set up a proper relational representation for CILP++ and thus direct comparison remains as a part of our future work.

[5] I.e., the template does not relate to toxicity or any other specific property of molecules and might be as well used for other classification tasks, too.

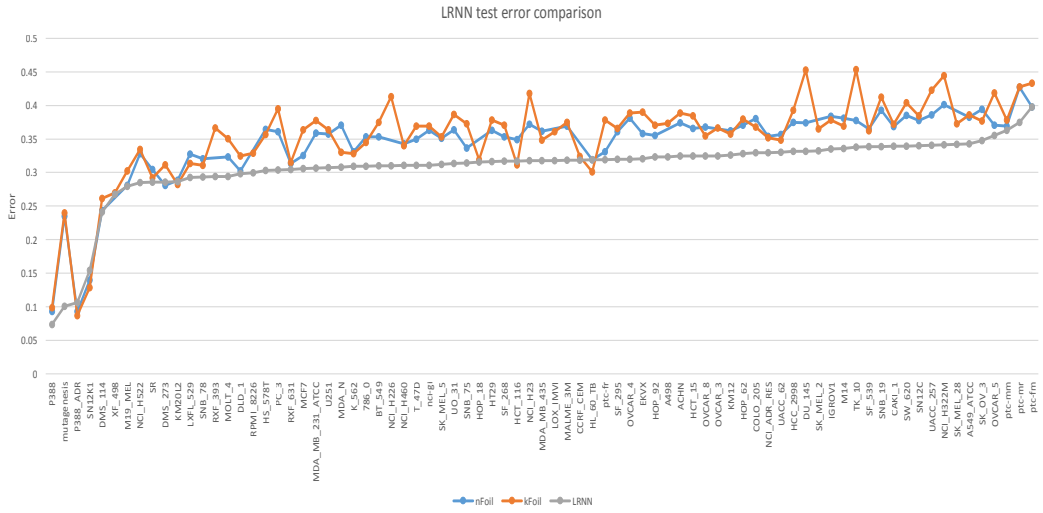

Figure 4: Prediction errors of LRNNs, kFOIL and nFOIL measured by cross-validation on 78 datasets of organic molecules.

In order to test the discovery of latent relational concepts, we performed an additional experiment with the Mutagenesis dataset. The relational concepts we were interested in were chains of varying lengths (up to 5 atoms). We trained the resulting LRNN to optimize the template's weights, however here we were more interested in extracting the learned patterns. We determined the chains of atoms which gave the highest output for the learned latent predicates. We obtained the following atom chain structures: C-C-F, N-O, C-Cl, C-Br, C-C-O, O-N-C. At least some of these structures appear to be directly relevant for the mutagenicity as they contain organic structures containing halogen atoms (Br, F and Cl). The other structures may be relevant to mutagenicity in combination with other structures.

Further, we have also investigated and confirmed the capability of LRNNs to learn proper weights of the latent non-ground relational concepts. This can be demonstrated e.g. on soft clustering of FOL predicates, a demonstration of which can be found in [20], together with evaluation and a closer description of the latent relational concept learning with Lifted Relational Neural Networks.

6 Conclusions

In this paper, we have introduced a method combining relational-logic representations with feedforward neural networks. The introduced method is close in spirit to lifted graphical models as it can be viewed as providing a lifted template for construction of ground neural networks. The performed experiments indicate that it is possible to achieve state-of-the-art predictive accuracies by weight learning with very generic templates and that it is able to induce notable latent relational concepts. There are many directions for future work including structure learning, transfer learning or studying different collections of activation functions. An important future direction is also the question of extending LRNNs to support recursion.

Acknowledgments

GS and FŽ are supported by Cisco sponsored research project "Modelling network traffic with relational features". While with CTU, OK was supported by the Czech Science Foundation through project P202/12/2032 and now by a grant from the Leverhulme Trust (RPG-2014-164).

References

[1] Sebastian Bader and Pascal Hitzler. Dimensions of Neural-symbolic Integration - A Structured Survey. *arXiv preprint*, page 28, 2005.

[2] H Blockeel and W Uwents. Using neural networks for relational learning. In *ICML-2004 Workshop on Statistical Relational Learning and its Connection to Other Fields*, 2004.

[3] M Botta, Giordana A, and R Piola. Combining first order logic with connectionist learning. In *Proceedings of the 14th International Conference on Machine Learning*, 1997.

[4] Artur S. d'Avila Garcez, Krysia Broda, and Dov M. Gabbay. *Neural-Symbolic Learning Systems: Foundations and Applications*. 2012.

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

[6] Manoel VM França, Gerson Zaverucha, and Artur S dAvila Garcez. Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine learning*, 94(1):81–104, 2014.

[7] Christoph Helma, Ross D. King, Stefan Kramer, and Ashwin Srinivasan. The predictive toxicology challenge 2000–2001. *Bioinformatics*, 17(1):107–108, 2001.

[8] Steffen Hölldobler, Yvonne Kalinke, and Hans Peter Störr. Approximating the semantics of logic programs by recurrent neural networks. *Applied Intelligence*, 11(1):45–58, 1999.

[9] Kristian Kersting and Luc De Raedt. Towards combining inductive logic programming with bayesian networks. In *Inductive Logic Programming, 11th International Conference, ILP 2001, Strasbourg, France, September 9-11, 2001, Proceedings*, pages 118–131, 2001.

[10] A Kimmig, L Mihalkova, and L Getoor. Lifted graphical models: a survey. *Machine Learning*, 99(1):1–45, 2015.

[11] George Klir and Bo Yuan. *Fuzzy sets and fuzzy logic*, volume 4. Prentice Hall New Jersey, 1995.

[12] Mark-A Krogel, Simon Rawles, Filip Železný, Peter A Flach, Nada Lavrač, and Stefan Wrobel. *Comparative evaluation of approaches to propositionalization*. Springer, 2003.

[13] N. Landwehr, A. Passerini, L. De Raedt, and P. Frasconi. kFOIL: learning simple relational kernels. In *AAAI'06: Proceedings of the 21st national conference on Artificial intelligence*, pages 389–394. AAAI Press, 2006.

[14] Niels Landwehr, Kristian Kersting, and Luc De Raedt. Integrating naive bayes and foil. *The Journal of Machine Learning Research*, 8:481–507, 2007.

[15] Huma Lodhi and Stephen Muggleton. Is mutagenesis still challenging. *ILP-Late-Breaking Papers*, 35, 2005.

[16] Liva Ralaivola, Sanjay J. Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural Netw.*, 18(8):1093–1110, 2005.

[17] J Ramon and L De Raedt. Multi instance neural networks. In *Proceedings of the ICML Workshop on Attribute-Value and Relational Learning*, 2000.

[18] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

[19] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 20(1):61–80, January 2009.

[20] Gustav Sourek, Vojtech Aschenbrenner, Filip Zelezny, and Ondrej Kuzelka. Lifted Relational Neural Networks. *arXiv preprint*, 2015.

[21] Geofrey G Towell, Jude W Shavlik, and Michiel O Noordewier. Refinement of approximate domain theories by knowledge-based neural networks. In *Proceedings of the eighth National conference on Artificial intelligence*, pages 861–866. Boston, MA, 1990.