Revisiting direct neuralisation of first-order logic

Iain A. D. Gunn
Department of Computer Science
Middlesex University
London
Email: i.gunn@mdx.ac.uk

David Windridge
Department of Computer Science
Middlesex University
London
Email: d.windridge@mdx.ac.uk

***Abstract*—There is a long history of direct translation of propositional Horn-clause logic programs into neural networks. The possibility of translating first-order logical syntax in the same way has been largely overlooked, perhaps due to a "propositional fixation" fixation! We briefly revise the possibility and advantage of translating existentially and universally quantified clauses into a neural form that follows the first-order syntax in a natural way.**

I. Introduction

The direct translation of logic programs into artificial neural networks has a relatively long history. A standard approach to neuralisation of Horn clauses, using a local representation in which each (ground) atom corresponds to a single dedicated neuron, is exemplified by the Knowledge-Based Artificial Neural Network of Towell and Shavlik [1].

Networks of this type have been criticised as having a "propositional fixation" [2]: a finite neural network can represent only a finite number of ground atoms, and can therefore represent a logic program only for a finite base. A language with first-order syntax but only a finite alphabet of symbols is equivalent to propositional logic, because any universally quantified ("for all X") clause can be translated into finitely many propositional clauses of the same form, one for each possible value of X. Thus, networks of the KBANN type cannot implement "true" first-order logic programs, only a finite fragment of first-order logic. [1]

However, in practice the syntax of first-order logic may be very valuable, notwithstanding this formal equivalence to propositional logic. Making a thousand or a million copies of a clause, one for each possible grounding of a variable X, might under some circumstances be desirable behaviour for a compiler creating machine code for immediate use, but is very undesirable under other circumstances. One such circumstance is if human interpretability is desired, but even if the rule-base is to be refined purely automatically, it may well be desirable to maintain the concept that there is a universal rule, rather than a thousand unconnected rules, each of which might be only very weakly evidentially supported on its own (This is the thinking behind the "connectionist" but non-neural Markov Logic Network approach [5]).

[1]Note, though, that Hölldobler et al. [3] give a neural method that will approximate, in a sense, the immediate-consequence operator T_P of a true First-Order Logic program, to a desired degree of accuracy in a real embedding. This is discussed in section 6 of the recent Besold et al. survey [4]

Most existing literature on neural translation of logic programs (with local representation) focuses on propositional logic. We believe the question of the translation of first-order syntax may have been neglected because of the well-known "propositional fixation", despite the possible practical advantages of first-order syntax even in the case of a finite base. Often, authors offhandedly mention that their architecture, designed with propositional logic in mind, is also in principle applicable to finite fragments of first order logic, provided they are fully grounded. Although such statements are true in their broad thrust, a lot of details may be hidden by the simple word "grounding", in that a naïve exhaustive approach to translating a first-order logic program to grounded propositional statements may lead to a very inefficient neural structure.

For this reason, we seek in the present paper to review and explore some approaches to neuralisation of logic programs with first-order syntax which may offer advantages over the naïve approach.

II. Related non-neural approaches

Closely related "connectionist" approaches to first-order logic are the Connectionist Horn Clause Logic system of Hölldobler and Kurfess [6], and the MMCILP model of Hallack et al. [7]. Hallack et al. follow the neural topology of KBANN/CILP, but use a generalised concept of neuron based on the "discrete neuron" of Sun [8].

Markov Logic Networks [5] combine First-Order Logic rules with the inference techniques of probabilistic graphical models, rather than neural methods. MLNs are discussed in section 7 of the recent comprehensive survey by Besold et al. [4].

III. Translation for existentially and universally quantified clauses

Consider first the case where some predicates cannot simultaneously be true of certain pairs of groundings, and this exclusion principle can be encoded into the logic program in a systematic way, using first-order syntax, as an existentially and universally quantified clause. Such a clause can be translated into a neural structure in a way that naturally follows the quantification syntax, as in the First-Order Neural Network of Botta et al. [9], with potentially great advantage over

naïvely translating the first-order clause into many propositional clauses. Botta et al. write: "even when a problem can be reduced to a propositional setting, the solutions found in FOL are more abstract and simpler than the corresponding ones in propositional logics". The following example will illustrate this.

Consider the case where a variable, I, represents an object ID. Consider a predicate $C(I, X, T)$, representing the concept that the object with identity code I is in location X at time T. Then, for a suitably sharply defined concept of location, it can never be the case that $C(i_1, x_1, t_1) = C(i_1, x_2, t_1)$ for $x_1 \neq x_2$. If each object is required to occupy one and only one location at all times, this principle could sensibly be incorporated into the logic program as a single-predicate clause:

$$\forall I, T \quad \exists! X \quad C(I, X, T). \tag{1}$$

To use a KBANN-style approach to neuralisation, it is necessary to first translate this first-order rule into many propositional rules:

$$NOT \quad (\ C(i_1, x_1, t_1) \quad AND \quad C(i_1, x_2, t_1)\ ), \tag{2}$$

and many other rules of this form (and some further OR clauses to enforce existence as well as uniqueness). Thus, a naïve KBANN-style neural translation of (1) would involve creating an input layer with a neuron for every possible combination of groundings of I, X, and T, and then a second layer with a neuron connected to each pair of input neurons with the same values of I and T, to check that no two such neurons fire at once. This neural translation is illustrated in Figure 1.

If there are n_i possible values of I, n_x possible values of X, and n_t possible values of T, this neuralisation requires $n_i n_x n_t$ neurons in the input layer, and $n_i n_t n_x (n_x - 1)$ neurons in the second layer. Even for a relatively small system with $n_i = 10$ object IDs, $n_t = 10$ time slots, and $n_x = 10$ possible object locations, this results in ~ 10000 neurons in the hidden layer, just to implement the single exclusivity clause. (And yet more neural wiring would be required to enforce existence of a suitable x for each (i, t): the network illustrated in Figure 1 implements only uniqueness.)

This combinatorial explosion in the number of neurons is very undesirable. But a better neural translation is possible through *following* the first-order syntax of (1), instead of removing it through immediate propositionalistaion. This is illustrated in Figure 2.

The first (left-most) layer of neurons in Figure 2 is the same as for Figure 1: it consists of all $n_i n_x n_t$ possible groundings of $C(I, X, T)$, arranged in $n_i n_t$ groups of n_x neurons each, giving all possible groundings of X for each choice of groundings for (I, T). But in the second layer, where for each such group in Figure 1 there was a neuron for each choice of pairing of neurons in the first-layer group, in Figure 2 there is now a *single* neuron for each group. Thus, the neuronal

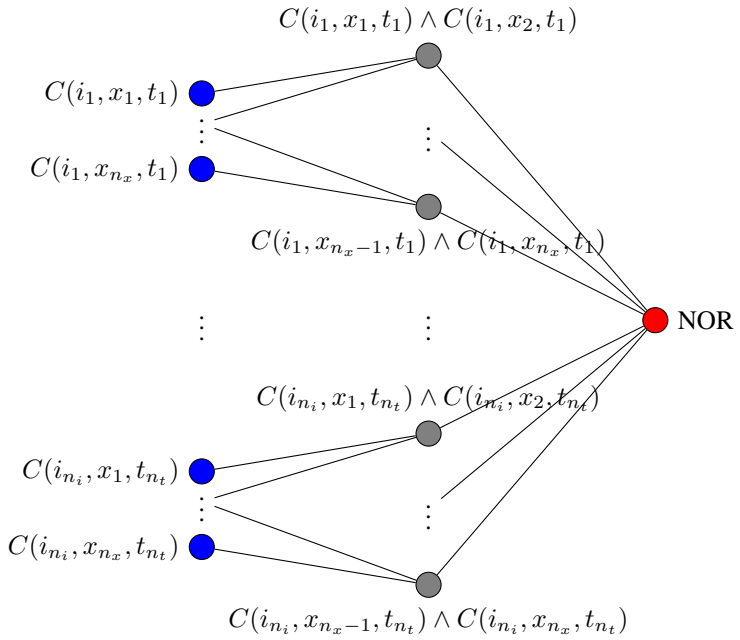

Fig. 1. Naïve neural implementation of uniqueness requirement of clause (1).

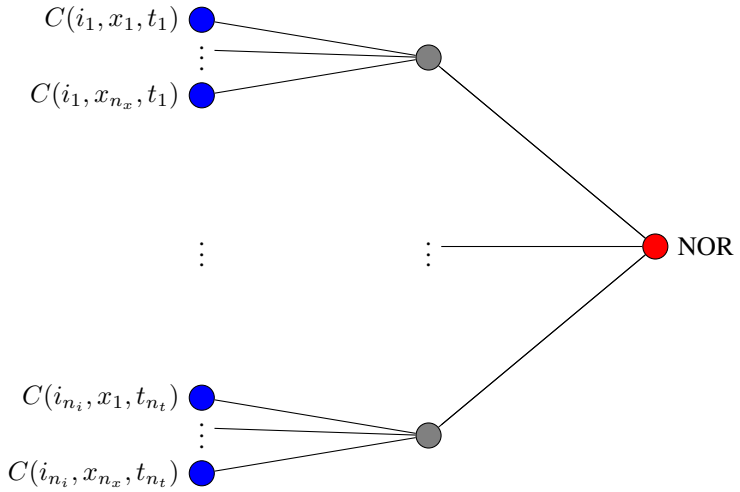

Fig. 2. Improved neural translation of (1). The threshold of each grey neuron is set such that it fires if more than one of the blue input neurons it is connected to fires.

budget for Figure 2 is dominated by the $n_i n_x n_t$ neurons in the input layer. (As with Figure 1, the neural network depicted enforces only uniqueness and not existence, but the extra neural architecture to enforce existence is small compared with the size of the input layer, and identical in each of the cases Fig. 1 and Fig. 2.) Thus, in the example we considered with $n_i = n_x = n_t = 10$, the neural approach of Fig. 2 requires ~ 1000 neurons, an order of magnitude (in general, a factor of n_x) less than the approach of Fig. 1.

IV. Facts

Other improvements are possible over the mechanical process of neuralising a propositionalised form of a logic program. The lowest-hanging fruit might be assertions of fact. A logic program might assert that some groundings of some predicates are always true: these may be called "facts" of the program. In a naïve neuralisation, these groundings would be represented by neurons that are constrained always to fire. But in some cases there is scope for producing a much reduced neural translation of the program, by introducing only a little intelligence into the grounding process.

Consider the program

D(X,Y) :- C(X),S(X,Y);

S(1,2);

S(2,1);

where both X and Y take values in $\{1, 2\}$.

The "full-grounding" approach to the neuralisation of this toy program is illustrated in Figure 3.

With the very least addition of sophistication to the process of exhaustively grounding the terms, it may be determined that groundings containing facts about S can be simplified: D(1,2) :- C(1),S(1,2), combined with the assertion that S(1,2) is true, reduces simply to D(1,2) :- C(1). (See figure 4). Any other clauses depending on the same fact could be simplified in the same way.

This takes us the first step down the path of automated theorem proving, and less trivial improvements might be gained through use of an intelligent "neural compiler" able to streamline a given program using the techniques of theorem-proving. This is a way of using a theorem-prover that allows the online system for inference to be fully neuralised, rather than using a logical system online as part of a hybrid system.

V. Discussion

The form of the network in Figure 2 is reminiscent of the Convolutional Neural Networks familiar from image-recognition applications. The same neural structure, in this case implementing uniqueness, is repeated across multiple blocks of input neurons. This raises the possibility of parameter-sharing when training such a network, as for CNNs, thus preserving the idea of a single first-order rule even following the grounding and neural translation.

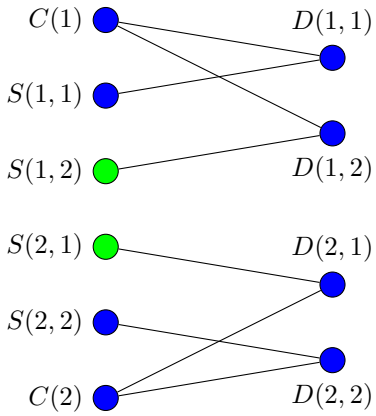

Fig. 3. Naïve neuralisation of example program from section IV

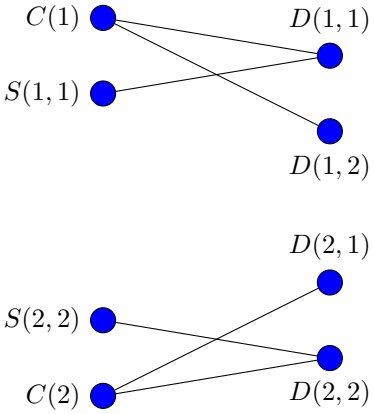

Fig. 4. Improved neuralization of example program from section IV

Acknowledgment

This work was supported by the EU 2020 project Dreams4Cars, grant number 731593

References

[1] G. Towell and J. W. Shavlik, "Knowledge based artificial neural networks," *Artificial Intelligence*, vol. 70, no. 4, 1994.

[2] J. McCarthy, "Epistemological challenges for connectionism," *Behavioral and Brain Sciences*, vol. 44, 1988.

[3] S. Hölldobler, F. Kurfess, and H.-P. Störr, "Approximating the semantics of logic programs by recurrent neural networks," *Applied Intelligence*, vol. 11, no. 1.

[4] T. R. Besold, A. d'Avila Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kühnberger, L. C. Lamb, D. Lowd, P. M. V. Lima, L. de Penning, G. Pinkas, and G. Zaverucha, "Neural-symbolic learning and reasoning: a survey and interpretation," *arXiv preprint arXiv:1711.03902*, 2017.

[5] M. Richardson and P. Domingos, "Markov logic networks," *Machine Learning*, vol. 62, no. 1–2, 2006.

[6] S. Hölldobler and F. Kurfess, "CHCL - a connectionist inference system," 1992.

[7] N. A. Hallack, G. Zaverucha, and V. C. Barbosa, "Towards a hybrid model of first-order theory refinement," *Hybrid Neural Systems*, 2000.

[8] R. Sun, "Robust reasoning: integrating rule-based and similarity-based reasoning," *Artificial Intelligence*, vol. 75, 1995.

[9] M. Botta, A. Giordana, and R. Piola, "FONN: Combining first order logic with connectionist learning," *ICML*, 1997.