ILP Operators for Propositional Connectionist Networks
Extended Abstract

Miguel Angel Gutiérrez-Naranjo* and Pascal Hitzler

[1] Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
magutier@us.es
[2] Institut für Künstliche Intelligenz, Fakultät für Informatik
Technische Universität Dresden
phitzler@inf.tu-dresden.de

1 Introduction

The study of the integration of symbolic logic and connectionist systems is an active area of research. Its general objective is appealing: Since biological neural networks are able to process symbolic information, it should be possible to do the same with artificial ones. This research should, so one of the visions, also shed light on the way in which neural networks represent symbolic information. Hence, researchers have set out to study the interplay between logic and connectionist systems, however with rather limited success so far, both on the theoretical and on the applied side, see [BS01] for a recent survey. It is conjectured [Höl00], that entirely new methods need to be developed in order to obtain satisfactory theoretical or practical results. In this extended abstract, we present preliminary results which relate inductive logic programming and connectionist systems. To the best of our knowledge, such a relation has not been studied before.

One approach to learning refered to in the literature on logic and connectionist systems proceeds along the following three steps [dGZ99]: (1) Insertion of knowledge into an artificial neural network, (2) training of this network with one of the standard learning algorithms and (3) extraction of rules from the trained network. This way, it is attempted to achieve learning of logical rules via learning algorithms which act on connectionist systems, i.e. artificial neural networks.

In this paper, we take a different perspective. In the inductive logic programming (ILP) paradigm, learning operators have been developed which perform learning on sets of logical rules in the form of logic programs. Since the tasks of knowledge insertion into and rule extraction from certain kinds of connectionist networks are fairly well understood, we can also carry over ILP operators to connectionist paradigms, thus obtaining learning operators on these systems. This

* The first named author acknowledges support by the International Quality Network "Rational mobile agents and systems of agents" at Technische Universität Dresden, funded by the DAAD.

transformation of ILP learning operators to operators which act on connectionist networks will be undertaken in the sequel. We will display the technique for the "identification" operator, and further results will be contained in the full version of the paper.

2 Preliminaries

A *3-layer feedforward artificial neural network with threshold activation* consists of sets V, C, Y of *nodes* or *units*, *weight functions* (or *weights*) $w_I : V \times C \to \mathbb{R}$, $w_O : C \times Y \to \mathbb{R}$, *threshold functions* (or *thresholds*) $t_O : Y \to \mathbb{R}$, $t_H : C \to \mathbb{R}$, and a *threshold activation function* $\sigma : \mathbb{R} \to \{0, 1\}$ with $\sigma(x) = 0$ for $x < 0$ and $\sigma(x) = 1$ for $x \geq 0$. V is called the *input layer*, Y is called the *output layer*, and C is called the *hidden layer* of the network.

Units are considered to hold the numbers 0 or 1, which we call *activations*, where "1" stands for "active" and "0" for "inactive". Computation is performed by propagating values through the network, from the input to the output layer. The activation of a unit y_k in the output layer is calculated by

$$y_k = \sigma\left(\sum_{c_j \in C} w_O(c_j, y_k) \cdot \sigma\left(\sum_{v_i \in V} w_I(v_i, c_j) v_i - t_H(c_j)\right) - t_O(y_k)\right) \qquad (1)$$

A *propositional (normal) logic program* is a finite set of propositional *clauses* or *rules* of the form $p \leftarrow q_1, \ldots, q_n$ where p is an atom (i.e. a propositional variable), q_i, for each i, is either an atom or a negated atom, and the commata stand for conjunctions. We call p the *head* of the clause and $q_1, \ldots, q_n$ the *body* of the clause. We impose the mild condition that for any atom p occuring in the body of a clause, $\neg p$ does not occur in the same clause. For our purpose, we can abstract from the order of the literals in the bodies of clauses. A *definite* propositional program is a propositional program in which no negation occurs.

3 Propositional Networks

In order to formalize representations of logic programs by connectionist systems, we will use certain kinds of three-layer feedforward neural networks.

Definition 1 (Propositional Network (PN)). *A* propositional network (PN) *is a septuple* $(V, C, w, w_I, w_O, t_H, t_O)$ *consisting of two finite sets* V *and* C*, a real number* w*, and four functions* $t_O : V \to \mathbb{R}$, $t_H : C \to \mathbb{R}$, $w_I : V \times C \to \{-w, 0, w\}$ *and* $w_O : C \times V \to \{0, w\}$.

For a given propositional network and $c \in C$, we define the following sets.

$$c_+ = \{v \in V : w_I(v, c) = w\} \qquad c^+ = \{v \in V : w_O(c, v) = w\}$$
$$c_- = \{v \in V : w_I(v, c) = -w\}$$

We can interpret a propositional network as a three-layer feedforward network with binary threshold activation functions in the following way: The input and ouput layers each consist of a copy of V. The hidden layer consists of a copy of C. The functions t_O, and t_H, respectively, yield the thresholds for the output and hidden layer, respectively. The functions w_I and w_O yield the weights between the input and hidden, respectively the hidden and output layer. Propagation in the network is via weighted sums, as usual, following equation (1) with threshold activation function σ. We will transform propositional logic programs into propositional networks using a transformation due to [HK94]. After such a transformation V will represent the set of propositional variables in the program and C will represent the set of clauses in the program.

Definition 2 (Propositional Program Network ($\mathrm{PPN_t}$)). *A* propositional program network with binary threshold activation function ($\mathrm{PPN_t}$) *is a propositional network in which t_O is constant with value $\frac{1}{2}$, $w = 1$, $|c^+| = 1$, and $t_H(c) = |c_+| - \frac{1}{2}$ for all $c \in C$. A* definite propositional network with binary threshold activation function ($\mathrm{DPN_t}$) *is a* PPN_t *where w_I maps to* $\{0, 1\}$ *only.*

Note that every quadruple (V, C, w_I, w_O) gives rise to a $\mathrm{PPN_t}$, i.e. to a septuple $(V, C, w, w_I, w_O, t_H, t_O)$, by setting $w = 1$, $t_O = \frac{1}{2}$, and computing t_H as in Definition 2, provided that $|c^+| = 1$ for all $c \in C$. We next recall the transformation due to [HK94]. In its original form, it was stated as an algorithm.

Transformation 1 *Let P be a propositional (normal) logic program. Let $V = \{v_1, \ldots, v_m\}$ be the set of all propositional variables occuring in P and let $C = \{c_1, \ldots, c_n\}$ be set set of all clauses in P. The* PPN_t associated with P *is the quadruple (V, C, w_I, w_O), where $w_I(v, c) = 1$, respectively $w_I(v, c) = -1$, if v appears as a body atom, repectively as a negated body atom, in c, and $w_I(v, c) = 0$ otherwise, and $w_O(c, v) = 1$ if c has head v, and $w_O(c, v) = 0$ otherwise. The resulting* PPN_t *is denoted by* $\mathrm{PPN}_t(P)$.

The intuition underlying the construction in Transformation 1 is, that to each program a network is associated whose input-output function captures the immediate consequence operator of the program (see [HK94]).

Proposition 1. *For every* PPN_t *N there exists a unique program P such that* $\mathrm{PPN}_t(P) = N$*, and for every* DPN_t *N there exists a unique definite program P with* $\mathrm{PPN}_t(P) = N$.

4 ILP Operators for Propositional Networks

We selected operators of the propositional ILP system DUCE [Mug87] for the transfer to propositional program networks, more specifically, identification, absorption, intra-construction, inter-construction, and truncation. We will display the technique on the identification operator, and the others will be covered in the full version of the paper.

Identification:

$$\frac{p \leftarrow A, B \quad p \leftarrow A, q}{q \leftarrow B \quad\; p \leftarrow A, q}$$

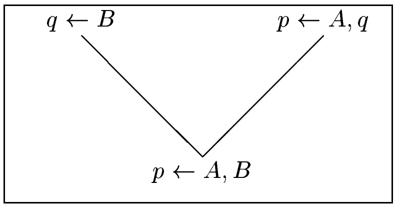

Fig. 1. Identification

Identification is a so-called V-operator, i.e. it is based on the idea of inverting resolution. In Duce, the identification operator takes as input two clauses $p \leftarrow A, B$ and $p \leftarrow A, q$ and yields the two clauses $p \leftarrow A, q$ and $q \leftarrow B$, see Figure 1, where p, q are propositional variables and A, B are conjunctions of propositional variables.

We can translate identification to $\mathrm{DPN_t}$s and $\mathrm{PPN_t}$s as follows, where $\uplus$ denotes disjoint union.

Operator 1 (Identification on $\mathrm{DPN_t}$s) *Let $c, d \in C$, $p, q \in V$, and $A, B \subseteq V$ be such that the following conditions hold:*

(I1) $p \in c^+ \cap d^+$.
(I2) $d_+ = \{q\} \uplus A$ *and* $c_+ = A \uplus B$.

Then the transformed network is the same quadruple as the original one, but for the weights (the mappings w_O and w_I), which have to be adjusted as follows.

- *Set to* 0*:* $w_I(a, c)$*, for all* $a \in A$ *and* $x_O(c, p)$
- *Set to* 1*:* $w_O(c, q)$

Operator 2 (Identification on $\mathrm{PPN_t}$s) *Let $c, d \in C$, $v, x \in V$, A, A', B and B' subsets of V be such that (I1), (I2) and (I3) hold.*

(I3) $d_- = A'$ *and* $c_- = A' \uplus B'$.

Then the transformed network is the same quadruple as the original one, but for the weights, which have to be adjusted as follows.

- *All changes listed in Operator 1.*
- *Also set to* 0*:* $w_I(a, c)$ *for all* $a \in A'$.

The network transformation is depicted in Figure 2. The following two results follow easily from the definitions.

Proposition 2. *Let P be a propositional logic program and let Q be a program obtained from P by applying identification. Then one can apply identification to $\mathrm{PPN}_t(P)$ in order to obtain $\mathrm{PPN}_t(Q)$.*

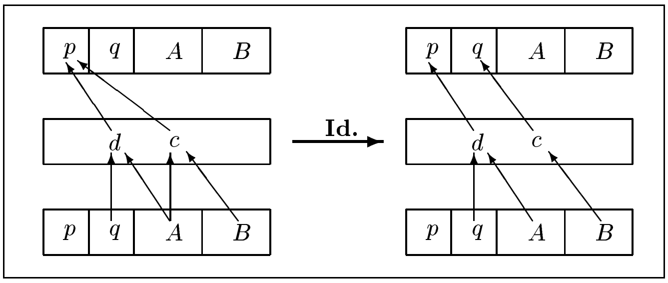

Fig. 2. Scheme for identification

Proposition 3. *Let $M, N \in \mathrm{PPN}_t$ such that N can be obtained from M by applying identification. For $M = \mathrm{PPN}_t(P)$ and $N = \mathrm{PPN}_t(Q)$, we have that Q can be obtained from P via identification.*

The other V-operator of the DUCE system, absorption, can be treated similarly. Like identification, it leaves the number of nodes in the layers unchanged. The operators intra-construction and inter-construction, which are also called W-operators since they can be understood as a combination of two V-operators, and truncation, differ from the V-operators in that their connectionist representation changes the number of nodes in the network. More specifically, inter- and intra-construction add nodes to all layers, while truncation removes nodes from the hidden layer. As already noted, details will be contained in the full version of the paper. For all the operators, results corresponding to Propositions 2 and 3 hold.

5 Conclusions and Further Work

We have presented some first observations on relating inductive logic programming and connectionist systems. Starting from these observations, it can be attempted to develop a connectionist inductive learning system based on ILP operators. By the nature of the identification operator as depicted in Figure 2, it should be noted that such a learning system will necessarily be based on a recurrent architecture, with recursive links from the output to the input layer, as used in [HK94].

This said, it is straightforward to extend our approach to networks with sigmoidal activation functions as in [dGZ99]. This makes it possible, in principle, to combine ILP learning operators, or more general operators of the same spirit, with backpropagation or other learning algorithms based on gradient descent.

Whenever continuous activation functions are involved, the casting of ILP operators into operators which transform networks is also related to a "fuzzification" of the underlying logic. Connections with work on Fuzzy ILP, as e.g. in [SIK$^+$99], or on quantitative logic programming, as in [Voj01], may guide the developments.

Other possible lines of extending the investigations concern generalizations towards predicate logic, e.g. along the lines of the SHRUTI system [SA93], which can deal with variable binding but is restricted to a subset of Datalog, i.e. first-order logic programs without function symbols. Attempts to represent first-order terms with function symbols in connectionist systems include the RAAM family, see [AD99], and topological approaches, see [HS0x], though it remains unclear at this stage how to carry over ILP operators to these settings.

References

[AD99] M.J. Adamson and R.I. Damper. B-RAAM: A connectionist model which develops holistic internal representations of symbolic structures. *Connection Science*, 10(1):41–71, 1999.

[BS01] A. Browne and R. Sun. Connectionist inference models. *Neural Networks*, 14(10):1331–1355, 2001.

[dGZ99] A. S. d'Avila Garcez and G. Zaverucha. The connectionist inductive lerarning and logic programming system. *Applied Intelligence, Special Issue on Neural Networks and Structured Knowledge*, 11(1):59–77, 1999.

[HK94] S. Hölldobler and Y. Kalinke. Towards a massively parallel computational model for logic programming. In *Proceedings of the ECAI94 Workshop on Combining Symbolic and Connectionist Processing*, pages 68–77. ECCAI, 1994.

[Höl00] S. Hölldobler. Challenge problems for the integration of logic and connectionist systems. In F. Bry, U. Geske, and D. Seipel, editors, *Proceedings 14. Workshop Logische Programmierung*, volume 90 of *GMD Report*, pages 161–171. GMD, 2000.

[HS0x] P. Hitzler and A. K. Seda. A note on relationships between logic programs and neural networks. In P. Gibson and D. Sinclair, editors, *Proceedings of the Fourth Irish Workshop on Formal Methods (IWFM'00)*, Electronic Workshops in Comupting (eWiC). British Computer Society, 200x. To appear.

[Mug87] S. Muggleton. Duce, an oracle based approach to constructive induction. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 287–292. Morgan Kaufmann, 1987.

[SA93] L. Shastri and V. Ajjanagadde. From simple associations to systematic reasoning: A connetionist encoding of rules, variables and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences*, 16(3):417–494, 1993.

[SIK+99] D. Shibata, N. Inuzuka, S. Kato, T. Matsui, and H. Itoh. An induction algorithm based on fuzzy logic programming. In N. Zhong and L. Zhou, editors, *Proceedings of the Third Pacific-Asia Conference on Knowledge Discovery and Data Mining*, volume 1574 of *Lecture Notes in Artificial Intelligence*, pages 268–273. Springer-Verlag, April 1999.

[Voj01] P. Vojtas. Fuzzy logic programming. *Fuzzy Sets and Systems*, 124(3):361–370, 2001.