Applying the connectionist inductive learning and logic programmingsystem to power system diagnosis

Conference Paper · July 1997

DOI: 10.1109/ICNN.1997.611649 · Source: IEEE Xplore

3 authors, including:

Artur D'Avila Garcez
City, University of London
156 PUBLICATIONS **1,122** CITATIONS

SEE PROFILE

Applying the Connectionist Inductive Learning and Logic Programming System to Power System Diagnosis*

Artur S. d'Avila Garcez†
Department of Computing - Imperial College
180 Queen's Gate, SW7 2BZ, London - UK
aag@doc.ic.ac.uk

Gerson Zaverucha‡
COPPE / UFRJ - Programa de Engenharia de Sistemas e Computação
Caixa Postal: 68511, CEP: 21945-970, Rio de Janeiro - Brazil
gerson@cos.ufrj.br

Victor Navarro A. L. da Silva
CEPEL - Centro de Pesquisas de Energia Elétrica
Caixa Postal: 2754, CEP: 20001-970, Rio de Janeiro - Brazil
navarro@fund.cepel.br

Abstract

The Connectionist Inductive Learning and Logic Programming System, C-IL^{2}P, integrates the symbolic and connectionist paradigms of Artificial Intelligence through neural networks that perform massively parallel Logic Programming and inductive learning from examples and background knowledge. This work presents an extension of C-IL^{2}P that allows the implementation of Extended Logic Programs in Neural Networks. This extension makes C-IL^{2}P applicable to problems where the background knowledge is represented in a Default Logic. As a case example, we have applied the system for fault diagnosis of a simplified power system generation plant, obtaining good preliminary results.

Keywords: Hybrid Systems, Machine Learning, Neural Networks, Extended Logic Programming, Power Systems.

1: Introduction:

It is generally accepted that one of the main problems in building Expert Systems (which are responsible for the industrial success of Artificial Intelligence) lies in the process of knowledge acquisition, known as the "knowledge acquisition bottleneck". An alternative is the automation of this process through Machine Learning techniques.

Learning, one of the basic attributes of intelligent comportment, can be defined as the change of behavior motivated by changes in the environment in order to perform better in many knowledge domains [16]. Learning strategies can be classified as: learning from instruction, learning by deduction, learning by analogy, learning from examples and learning by observation and discovery [16]; the latter two are forms of inductive learning.

Various learning strategies could be integrated in an intelligent hybrid system, exploring the advantages that

* This work is part of the project ICOM from CNPq - ProTem3.
† The author is partially financially supported by CAPES.
‡ The author is partially financially supported by CNPq.

each one presents. Taking into account the evolution of computer science for massively parallel architectures [2], it is desirable that intelligent hybrid systems could be based on some artificial neural network's model. In [25] and [26] is empirically showed that neural networks using the backpropagation learning algorithm [22] are at least as effective as purely symbolic inductive learning systems [12, 18].

Notwithstanding, neural networks should be able to justify their decision making process in order to acquire the so called "explanation capability", and to use prior knowledge while learning, like the symbolic systems. Therefore, it is very important to outline tight correlation between symbolic knowledge and Artificial Neural Networks.

Toward this goal, in [11] Holldobler and Kalinke presented a massively parallel method for logic programming [13]. They have shown that for each logic program P there exists a three-layer feedforward neural network R with binary threshold units that computes T_P, the fixed point operator of P. If R is transformed into a recurrent network by linking the units in the output layer to the corresponding units in the input layer, then it always settles down in a unique stable state (given any input vector, that is, any interpretation) when P is an acceptable program (see [3])[1]. This stable state is the unique stable model of P, that is the least fixed point of T_P [6].

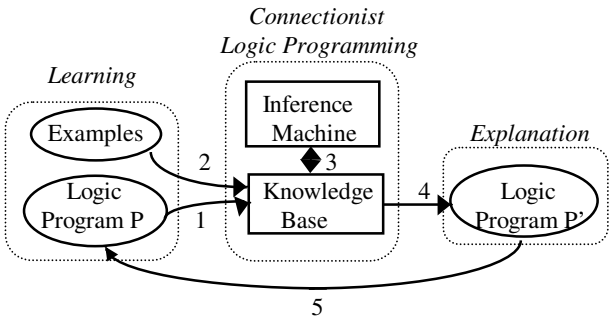
In [26] Towell and Shavlik presented KBANN[2], a system for rule insertion, refinement and extraction from neural networks. In KBANN, the background knowledge defines the neural network's initial topology through a rule inserting procedure, the backpropagation learning from examples algorithm is responsible for the network's refinement, and the extraction of rules for its explanation capability.

Considering the above results, the C-IL^{2}P system ("Connectionist Inductive Learning and Logic Programming System") [4, 5] integrates two learning strategies (deductive and inductive) in the same environment, a simple artificial neural network architecture, exploiting its best property - the inductive learning skill. Therefore, a single computational model can represent and process symbolic knowledge in a massively parallel way, performing learning from examples with background knowledge as in [26], and logic programming as in [11]. This integration also offers explanation capability to the system through the extraction of revised logic programs from the trained artificial neural network.

In this work, in section 2 we present an extension of the C-IL^{2}P system that allows the implementation of extended logic programming [7] in neural networks. Extended logic programming, that adds explicit negation to general logic programs, corresponds to a fragment of default logic [14]. In this way, the C-IL^{2}P system can be applied to problems in which the background knowledge is described in (this fragment of) default logic. As a case study, in section 3 we apply the system in fault diagnosis of a simplified power generator plant [23, 24], where the background knowledge is described using Poole's default logic system Theorist [20, 21]. In section 4, we conclude and discuss directions for future work.

2: The Extended C-IL^{2}P System

The C-IL^{2}P system is a purely connectionist, massively parallel system for logic programming, that is capable of performing inductive learning from examples and background knowledge.



(1)Background Knowledge Insertion.
(2)Connectionist Inductive Learning (Backpropagation) from examples and background knowledge.
(3)Connectionist Logic Programming System (Deduction).
(4)Revised Logic Program Extraction.
(5)Expert's validation and Feedback.

Figure 1. The C-IL^{2}P system.

Definition 2.1: A general logic program P is a set of clauses of the form $A \leftarrow L_1 ... L_n$, where $n \geq 0$, A is a atom and L_i $(1 \leq i \leq n)$ are literals.

The background knowledge, represented by a logic program P, is translated to an equivalent three-layer fully-

[1] It guarantees that *P* has exactly one stable model.
[2] Knowledge-based Artificial Neural Network.

connected neural network R, given by the Insertion Algorithm below [4, 5].

Insertion Algorithm: Let m and n be the number of literals and clauses occurring in P, respectively. Consider that the literals of P are numbered from 1 to m, such that the input and output layer of R are vectors of length m, where the i-th neuron represents the i-th literal of P.

1. Set the threshold (θ^I_i) of the neurons in the input layer to ½. In this way, the activation of the neurons in the input layer will represent the interpretations[3] for P, given by the input vector of R.

2. For each clause C_l of P of the form $A \leftarrow L_1, \ldots, L_k$:

   2.1. Add a neuron N^c_l to the hidden layer;

   2.2. Connect each neuron L_i ($1 \leq i \leq k$)[4] in the input layer to the neuron N^c_l in the hidden layer. If L_i is a positive literal then set the connection weight to W; otherwise, set the connection weight to -W, W > 0;

   2.3. Connect the neuron N^c_l in the hidden layer to the neuron A in the output layer and set the connection weight to W;

   2.4. Define the threshold (θ^H_l) of the neuron N^c_l in the hidden layer as (p - ½) W, where p is the number of positive literals occurring in the clause C_l.

3. For each fact F_l of P of the form $A \leftarrow$:

   3.1. Add a neuron N^f_l to the hidden layer;

   3.2. Connect the neuron T in the input layer to the neuron N^f_l in the hidden layer and set the connection weight to W, where T has threshold ½ and input data "1";

   3.3. Connect the neuron N^f_l in the hidden layer to the neuron A in the output layer and set the connection weight to W;

   3.4. Define the threshold (θ^H_l) of the neuron N^f_l in the hidden layer as W/2.

4. Set the threshold (θ^O_i) of the neurons in the output layer to W/2.[5]

[3] Truth assignments to the literals: "true" (input data "1") or false (input data "0").

[4] When we write "neuron L_j", it refers to the neuron in *R* that represents the literal L_j in *P*.

[5] Remember that *R* is fully-connected, thus all other connections have their weights fixed at zero.

Example 2.1: Consider the following program: P = { A ← B,C,~D; A ← E,F; B ← }. Using the insertion algorithm, we obtain the neural network R of Figure 2.

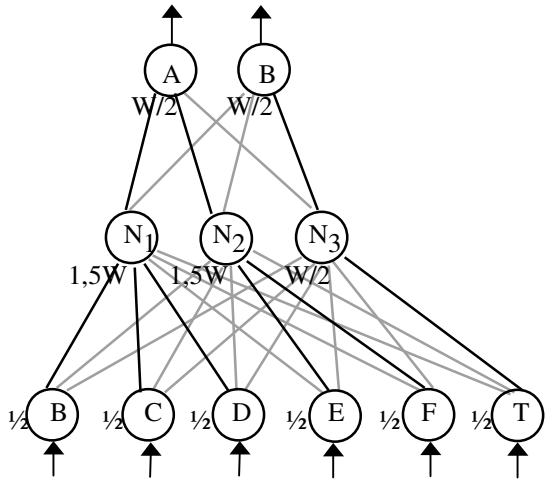


Figure 2. The Neural Network R obtained by the Insertion Algorithm over P.[6]

Definition 2.2: The meaning function T_P, mapping Herbrand interpretations (i) of P to Herbrand interpretations (I) of P is defined as follows: $T_P(i) = I = \{A \in B_P \mid$ there exists a ground instance of a clause in P of the form $A \leftarrow L_1 \ldots L_n$ such that i satisfies $L_1 \ldots L_n \}$.

Theorem 2.1 [4]: For each general logic program P, there exists a feedforward artificial neural network R with a positive W and exactly three layers of semi-linear neurons, obtained by the above "Insertion Algorithm", such that R computes T_P.

If R is transformed into a recurrent network by linking (with W = 1) the units in the output layer to the corresponding units in the input layer, we have:

Corollary 2.1 [4]: Let P be an acceptable program. There exists a recurrent neural network R such that, starting from an arbitrary input, R converges to the unique fixed-point of T_P.

Since we obtain a standard three-layer neural network after the insertion of the background knowledge, we can apply backpropagation in order to train it with examples. After that, the knowledge encoded in the trained neural

[6] The connections with weight zero are shown by the gray arcs. In order to perform inductive learning, these connections must receive random values near zero.

network R' can be extracted by the "extraction algorithm" [5] obtaining an equivalent and revised logic program P'.

Theorem 2.2 [5]: For each three-layer feedforward artificial neural network R' with semi-linear neurons there exists a general logic program P', obtained by the "Extraction Algorithm", such that R' computes $T_{P'}$.

In [7], Gelfond and Lifschitz developed the Extended Logic Programming, adding explicit negation to general logic programs. The semantics of these programs are given by their Answer Sets which are based on the stable model semantics for general logic programs. Extended Logic Programming corresponds to a fragment of Reiter's default logic [14] in the sense that answer sets correspond to extensions.

A direct way to extend C-IL^{2}P system to perform Extended Logic Programming is to make it deal with explicit negation in its insertion algorithm. To do so, we simply need to define neurons in the input and output layers of R that explicitly represent the negation of a given atom. In order to C-IL^{2}P compute a logic program, a concept A is represented by a neuron in the input and / or output layer of the neural network, whose weights indicates whether A is a positive or negative literal (negation as failure), that is, it differentiates A from ~A. For an extended logic program, we must also represent the concept ¬A (explicit negation) in the neurons, whose weights, now, differentiates ¬A from ~¬A.

Notice that the extension of C-IL^{2}P for dealing with explicit negation in an extended logic program must be seen as a simple renaming of negative literals (i.e.: ¬A) by a new positive literal (i.e.: A') (see [7]). After doing that, we can apply directly C-IL^{2}P's insertion algorithm. Furthermore, if the resulting program, after renaming, is acceptable then theorems 2.1 and 2.2 still hold.

3. Application in Power System Diagnosis

We apply C-IL^{2}P in power systems diagnosis. Toward this goal, we use as a case example a simplified Power System Generation Plant (see Figure 3). This system has two generators, two transformers with their respective circuit breakers, two buses (main and auxiliary) and two transmission lines also with their respective circuit breakers. Each transmission line has six associated alarms: breaker status (indicates whether it is open or not), phase over-current (shows that there was an over-current in the phase line), ground over-current (shows that there was an over-current in the ground line), timer (shows that there was a distant fault from the power plant generator), instantaneous (shows that there was a close-up fault from the power plant generator), and auxiliary (indicates that the transmission line is connected to the auxiliary bus).

Each transformer has three associated alarms: breaker status (indicates whether it is open or not), overloading (shows that there was a transformer overload) and auxiliary (indicates that the transformer is connected to the auxiliary bus).

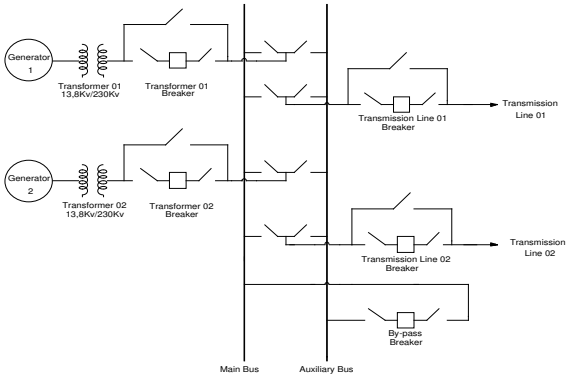


Figure 3. Configuration of a simplified Power System Generation Plant.

In the above electric system, we know that if the instantaneous alarm of transmission line 01 is activated there is a transmission line 01 close-up fault. This fact can be described in the following way:

Fact:

Alarm(instantaneous, transmission_line_01) ← Fault(close-up, transmission_line_01)

Hypothesis:

Fault(close-up, transmission_line_01)

The sentences above represent the knowledge in a default logic called Theorist [20]. Besides the background knowledge related to the usual faults and their associated alarms, we generate an example set that has noisy (absence of one of the characteristic alarms) single and multiple faults.

The aim of implementing the power system diagnosis in C-IL^{2}P is twofold: to use the background knowledge in the neural learning process and to explain the reasons for the activated alarm sets. For instance, if the set of three alarms (opening the breaker of the transmission line 01, transmission line 01 phase over-current and transmission

line 01 instantaneous) are activated, the system must diagnose the transmission line 01 phase-to-phase close-up fault.

Moreover, it is also possible that, in the case of a system fault, an alarm fail to activate due to some equipment failure. In order to handle this problem, our intelligent system should be able to detect faults even if the associated alarm set is not complete. For instance, if only phase over-current and instantaneous alarms of transmission line 01 are activated, the system must be able to diagnose the transmission line 01 phase-to-phase close-up fault, even without activation of the opening breaker alarm. As this fault is best explained by the two first alarms activated, it is reasonable that the intelligent system points out the fault occurrence. This characteristic motivates the use of artificial neural networks for the system diagnosis since they present good generalization performance in noisy situations (if sufficiently structured and trained).

In order to verify the system generalization ability, we applied two test cases: noisy (absence of one of the characteristic alarms) single faults (92 patterns) and noisy multiple faults (70 patterns). After submitting the total set of alarm patterns to the C-IL^{2}P neural network, the system achieved 97% of correctness.

Figure 4 shows the error reduction curves during the learning process using C-IL^{2}P and backpropagation neural networks. Notice that the faster convergence of C-IL^{2}P neural network is due to the use of the background knowledge in the learning process. This result confirms the importance of the background knowledge for inductive learning.

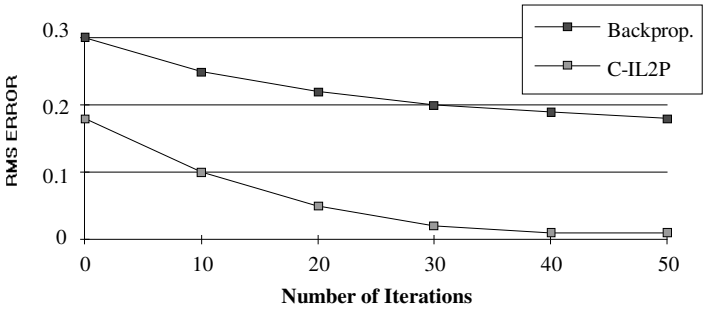


Figure 4. RMS Error in the learning process.

These preliminary results are considered to be very promising since they are similar to the ones described in [24] and show the good performance of C-IL^{2}P as an integrated system for inductive learning with background knowledge and default logic.

4. Conclusions and Future Work

In this work we presented an extension of C-IL^{2}P system that allows the implementation of Extended Logic Programs in Neural Networks. This extension makes C-IL^{2}P applicable to problems where the background knowledge is represented in a default logic. As a case example, we have applied the system for fault diagnosis of a simplified power system generation plant, obtaining good preliminary results.

These results indicates the usefulness of C-IL^{2}P as a tightly coupled hybrid system [9] for the solution of problems related to signal processing.

As future work, we would like to submit the application here presented to a more expressive quantity of tests, to apply C-IL^{2}P's extraction algorithm to this case study and to further examine the results.

5. References

[1] R. Andrews, J. Diederich and A. B. Tickle; "A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks"; Knowledge-based Systems; Vol. 8, n^o 6; 1995.

[2]] V. C. Barbosa; "Massively Parallel Models of Computation"; Ellis Horwood Series in Artificial Intelligence; 1993.

[3] M. Fitting; "Metric Methods - Three Examples and a Theorem"; Technical Report, CUNY - New York; 1993.

[4] A. S. Garcez, G. Zaverucha e L. A. Carvalho; "Logical Inference and Inductive Learning in Artificial Neural Networks"; Workshop on Neural Networks and Structured Knowledge, ECAI 96, Budapest; 1996.

[5] A. S. Garcez, G. Zaverucha, L. A. Carvalho; "Logic Programming and Inductive Learning in Artificial Neural Networks: Insertion, Extraction and Applications"; Technical Report, Coppe - Universidade Federal do Rio de Janeiro; 1996.

[6] M. Gelfond and V. Lifschitz; "The Stable Model Semantics for Logic Programming"; 5th Logic Programming Symposium, MIT Press; 1988.

[7] M. Gelfond and V. Lifschitz; "Classical Negation in Logic Programs and Disjunctive Databases"; New Generation Computing, Vol. 9, Springer-Verlag; 1991.

[8] P. J. Hayes; "In Defense of Logic"; 5th IJCAI; 1977.

[9] M. Hilario; "An Overview of Strategies for Neurosymbolic Integration"; Connectionist-Symbolic Integration: from Unified to Hybrid Approaches - IJCAI 95; 1995.

[10] S. Holldobler; "Automated Inferencing and Connectionist Models"; Postdoctoral Thesis, Intellektik, Informatik, TH Darmstadt; 1993.

[11] S. Holldobler and Y. Kalinke; "Toward a New Massively Parallel Computational Model for Logic Programming"; Workshop on Combining Symbolic and Connectionist Processing, ECAI'94; 1994.

[12] N. Lavrac and S. Dzeroski; "Inductive Logic Programming: Techniques and Applications"; Ellis Horwood Series in Artificial Intelligence; 1994.

[13] J. W. Lloyd; "Foundations of Logic Programming"; Springer - Verlag; 1987.

[14] W. Marek and M. Truszczynski; "Nonmonotonic Logic: Context Dependent Reasoning"; Springer-Verlag; 1993.

[15] L. Medsker; "Neural Networks Connections to Expert Systems"; Proceedings of the World Congress on Neural Networks; 1994.

[16] R. S. Michalski; "Learning Strategies and Automated Knowledge Acquisition"; Computational Models of Learning, Symbolic Computation, Springer-Verlag; 1987.

[17] M. Minsky; "Logical versus Analogical, Symbolic versus Connectionist, Neat versus Scruffy"; AI Magazine, Vol. 12, nº 2; 1991.

[18] S. Muggleton and L. Raedt; "Inductive Logic Programming: Theory and Methods"; The Journal of Logic Programming; 1994.

[19] G. Pinkas; "Reasoning, Nonmonotonicity and Learning in Connectionist Networks that Capture Propositional Knowledge"; Artificial Intelligence, Vol. 77; 1995.

[20] D. L. Poole, R. G. Goebel and R. Aleliunas; "Theorist: A Logical Reasoning System for Defaults and Diagnosis"; The Knowledge Frontier: Essays in the Representation of Knowledge, Springer-Verlag; 1987.

[21] D. L. Poole; "Compiling a Default Reasoning System into Prolog"; New Generation Computing; 1991.

[22] D. E. Rumelhart, G. E. Hinton and R. J. Williams; "Learning Internal Representations by Error Propagation"; Parallel Distributed Processing, Vol. 1, D. E. Rumelhart, J. L. McClelland and the PDP Research Group, MIT Press; 1986.

[23] V. N. A. L. da Silva, G. Zaverucha, Guilherme N. F. de Souza, "An Integration of Neural Networks and Nonmonotonic Logic for Power Systems Diagnosis", IEEE International Conference on Neural Networks, Australia; 1995.

[24] V. N. A. L. da Silva, Ricardo S. Zebulum, "An Integration of Neural Networks and Fuzzy Logic for Power Systems Diagnosis", Intelligent Systems Applications to Power Systems - ISAP'96; 1996.

[25] S. B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jong, S. Dzeroski, S. E. Fahlman, D. Fisher, R. Haumann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R. S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, K. Van de Welde, W. Wenzel, J. Wnek and J. Zhang; "The MONK's Problem: A Performance Comparison of Different Learning Algorithms"; Technical Report, Carnegie Mellon University; 1991.

[26] G. G. Towell and J. W. Shavlik; "Knowledge-Based Artificial Neural Networks"; Artificial Intelligence, Vol. 70; 1994.

[27] G. Zaverucha; "A Prioritized Contextual Default Logic: Curing Anomalous Extensions with a Simple Abnormality Default Theory"; Proceeding of the KI'94, LNAI 861, Springer-Verlag; 1994.

[28] G. Zaverucha; "On Cumulative Default Logic with Filters"; 6th International Workshop on Nonmonotonic Reasoning, Eds. M. Goldszmidt and V. Lifschitz; Timberline, Oregon, USA, pp.132-140; 1996.