Automated Expert Knowledge Base Generation Using Genetic Programming

Athanasios Tsakonas [a] [1] and Georgios Dounias [b]
[a] University of the Aegean, Dept. of Business Administration, 8 Michalon Str., Chios, Greece,
[b] University of the Aegean, Dept. of Finance, 31 Fostini Str., Chios, Greece

ABSTRACT: Nowadays, a large number of intelligent systems for decision-making have yielded encouraging results. However, it is commonly acknowledged that expert system technology has some drawbacks. In particular, the knowledge bases of expert systems do not evolve. To solve this problem, a solution that integrates neural networks and expert systems has recently proposed, namely the application of neural logic networks. This integrated system combines the strength of rule-based semantic structure and the learning capability of connectionist architecture. Nevertheless, the early approaches of this model carried the disadvantage of producing poor results or solutions that could not be interpreted straightforward. In this work, we overcome these problems and we propose a system that is capable of producing arbitrary large and connected neural logic networks that can easily be interpreted into sets of expert rules. To accomplish this task we adopt a genetic programming approach, guided through grammars and we encode indirectly the architecture into genetic programming individuals. We test and make conclusions on the effectiveness of the proposed system into two real-world decision making domains.

Keywords: Expert systems, Neural Logic Networks, Grammar Guided Genetic Programming, Cellular Encoding.

---

[1] Corresponding author. Tel: +306-937-891-399, Fax: +302-271-093-464
E-mail address: tsakonas@stt.aegean.gr (Athanasios Tsakonas)

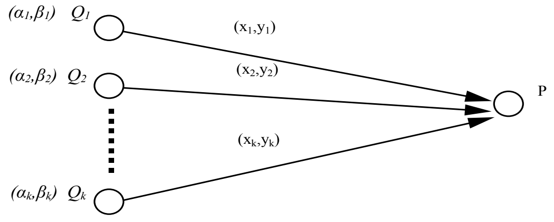
1. Introduction

Since their first application, artificial neural networks have been the research target for scientists that were aiming to obtain a meaningful to the humans interpretation of these networks' powerful structure. A specific class of artificial neural networks, so-called *neural logic networks,* by their definition can be easily interpreted in a number *of expert rules*. Hence, they can be considered as integration between rule-based expert systems and neural networks (Quah et al. 1996). However, the advantage in these networks is that they are not necessarily static as the expert systems, but can be trained according to the problem encountered. In (Teh 1995), a training methodology related to back-propagation was proposed. Later, the Supervised Clustering and Matching (SCM) algorithm (Tan and Teow 1997) was introduced. These training models however, aiming at the refinement of the edge weights often made the neural logic networks suffer in terms of their interpretability. This drawback led the research to alternative solving methodologies such as the genetic programming (Chia and Tan 2001). However, in their system, (Chia and Tan 2001) provided a system that was capable of producing only a limited number of neural logic network formats, that were those that resemble to a binary tree. In our work, we overcome these problems providing a framework for the production of neural logic networks and fuzzy neural logic networks that can be arbitrarily large and connected but still maintain their interpretability, and can be always translated in a series of expert rules. For this reason, we adapt a grammar guided genetic programming (Koza 1992) approach that uses cellular encoding (Gruau 1996) to describe the neural logic networks. Grammar-guided genetic programming for knowledge discovery is an extension to the original GP concept and it makes possible the efficient automatic discovery of empirical laws. It relates to the Machine Discovery framework, originally described by Langley (Langley et al. 1983), which incorporated inductive heuristics but suffered from limitations regarding ill-conditioned data and large search spaces (Ratle and Sebag 2000). Genetic programming however can easily overcome these problems mainly due to its stochastic nature. We apply this system to two real-world problems. The first problem is the NASA space shuttle landing control. Our results in this domain demonstrate the ability of the proposed system to explore easily understandable neural logic network representations and facilitate the knowledge discovery. The second problem we apply the system is the ionosphere radar data classification, where the system is proved capable of investigating and providing very complex neural logic network structures, however still maintaining the ability of interpretation of these networks into expert rules. The paper is organised as follows. In section 2 we introduce the theoretical background of the neural logic networks and the genetic programming framework. Section 3 contains the design and the implementation description of the proposed system. In section 4 we included the description of the problem domains, our system configuration and the obtained results together with a discussion. Finally, section 5 contains our conclusion regarding this work and proposed future directions for this domain.

2. Background

2.1 Neural Logic Networks

The neural logic network is a finite directed graph. It is usually consisted by a set of input nodes and an output node. In its 3-valued form, the possible value for a node can be one of three ordered pair activation values (1,0) for true, (0,1) for false and (0,0) for don't know. Every synapse (edge) is assigned also a an ordered pair weight (x,y) where x and y are real numbers. An example of a neural logic network and its output value (a,b) of node P is shown in *Fig. 2.1.*



(α_1,β_1) Q_1
(x_1,y_1)
(α_2,β_2) Q_2
(x_2,y_2)
(x_k,y_k)
(α_k,β_k) Q_k
P

$$(a,b)=\begin{cases}(1,0) & if\sum_{j=1}^{k}a_jx_j-\sum_{j=1}^{k}b_jy_j\geq 1\\ (0,1) & if\sum_{j=1}^{k}a_jx_j-\sum_{j=1}^{k}b_jy_j\leq -1\\ (0,0) & otherwise\end{cases}$$

Fig. 2.1. An example of a neural logic network and its output value.

Different sets of weights enable the representation of different logical operations. It is actually possible to map any rule of conventional knowledge into a neural logic network. In *Fig. 2.2*, a number of logical operators and their implementation in neural logic networks is shown. In this work, these operations, among others, are possible to be part of any candidate solution. According to (Teh 1995), the neural logic networks fulfill all the features that are required for the unification of the symbolic and neural processing. In the following example, we present an enhancement of the *PROLOG* programming language, using neural logic networks, aiming at a more powerful programming environment, the so-called "Neural Prolog" (Teh 1995). The procedure includes the following steps:

Step 1:We enhance the facts of PROLOG, by allowing to simple predicates to get three values: (1,0), (0,1) and (0,0). Hence,
Father(X,Y)=(1,0) means that it is true that X is Father of Y,

Father(X,Y)=(0,1) means that it is false that X is Father of Y,
Father(X,Y)=(0,0) means that it is still unknown if X is Father of Y

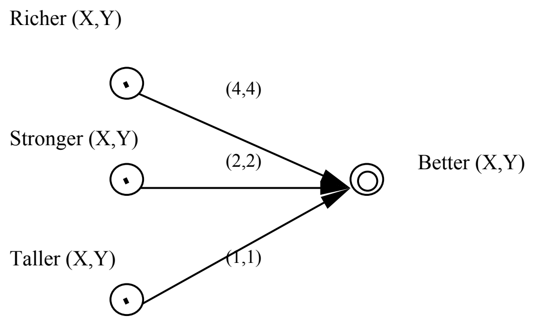


Fig. 2.2 Neural logic network example. The specific network corresponds to one priority rule

.
If in the same program appear both predicates Father(X,Y)=(1,0) and Father(X,Y)=(0,1), then the program is considered inconsistent. On the opposite, if the predicates Father(X,Y)=(1,0) and Father(X,Y)=(0,0) appear, then the program is consistent and the Father(X,Y)=(0,0) will be deleted. This generalisation can be expanded into fuzzy neural logic networks as well.
Step 2: We may create rules into the programming language PROLOG directly by every neural logic network. For example, the neural logic network shown in *Fig. 2.5*, which corresponds to a priority rule, produces a number of rules in PROLOG (If-Then rules), which are presented in *Fig. 2.3.*

```
If richer(X,Y)=(1,0) then better(X,Y)=(1,0)
If richer(X,Y)=(0,1) then better(X,Y)=(0,1)
Suppose richer(X,Y)=(0,0)
        If stronger(X,Y)=(1,0) then better(X,Y)=(1,0)
        If stronger(X,Y)=(0,1) then better(X,Y)=(0,1)
        Suppose richer(X,Y)=(0,0) and stronger(X,Y)=(0,0)
                If taller(X,Y)=(1,0) then better(X,Y)=(1,0)
                If taller(X,Y)=(0,1) then better(X,Y)=(0,1)
                        Suppose richer(X,Y)=(0,0) and stronger(X,Y)=(0,0)
                        and taller(X,Y)=(0,0) then better(X,Y)=(0,0)
```

Fig. 2.3. Rules in PROLOG that derive by the network of *Fig. 2.5*

From this example it is shown that every neural logic network may be interpreted into a set of *If-Then* rules of the PROLOG programming language, which is used mainly in artificial intelligence expert systems.

2.2 Grammar Guided Genetic Programming

Genetic Programming (GP) is a search methodology belonging to the family of evolutionary computation (EC). The prime advantage of genetic programming over classic genetic algorithms, is the ability to construct functional trees of variable length. This property enables the search for very complex solutions that are usually in the form of a mathematical formula - an approach that is commonly known as symbolic regression. When the syntax form of the desired solution is already known, it is useful to restrain the genetic programming from searching solutions with different syntax forms (Gruau et al. 1996, Montana 1995) with the help of a context-free grammar declared in the Backus-Naur-Form (BNF) (Gruau 1996, Janikow 1996, Naur 1963, Ryan et al. 1998). The BNF-grammar consists of terminal nodes and non-terminal nodes and is represented by the set {N,T,P,S} where N is the set of non-terminals, T is the set of terminals, P is the set of production rules and S is a member of N corresponding to the starting symbol. In order to explore variable sized solutions, usually a kind of indirect encoding is applied. The most common one is the cellular encoding (Gruau 1996), in which a genotype can be realized as a descriptive phenotype for the desired solution. More specifically, within such a function set, there are elementary functions that modify the system architecture together with functions that calculate tuning variables.

3. Design and Implementation

The GP parameters of the system are presented in *Table 1.* The selected parameters offered efficient runs throughout experiments..

Table 1 GP parameters for G^3P

Parameter	Value
Population:	2,000 individuals
GP implementation:	Steady-state G^3P
Selection:	Tournament with elitist strategy
Tournament size:	6
Crossover Rate:	0.35
Overall Mutation Rate:	0.65
Node Mutation Rate:	0.4
Shrink Mutation Rate:	0.6
Killing Anti-Tournament size:	2
Maximum allowed individualsize:	650 nodes
Maximum number of generations:	100

The system grammar is presented in *Fig. 3.1*. Initial symbol (root) of a tree can be only of a type <PROG>.

```
<PROG>        : =    PROG <PLACE1><SYNAPSE>
<PLACE1>      : =    S1 <PLACE1><SYNAPSE><PLACE2>
                     | P1 <PLACE1><PLACE1>
                     | IN
IN            : =    Data attribute (system input)
<PLACE2>      : =    S2 <PLACE2><SYNAPSE><PLACE2>
                     | P2 <PLACE2><SYNAPSE><PLACE2>
                     | E
E             : =    ∅
<SYNAPSE>     : =    LNK <NUM><CUT><SYNAPSE>
                     | CNR <CNRSEL><K>
<NUM>         : =    NUM
<CUT>         : =    CUT
<CNRSEL>      : =    CNRSEL
<K>           : =    K
NUM           : =    Integer in [1,256]
CUT           : =    Integer in [0,1]
CNRSEL        : =    Integer in [0,10]

K             : =    Integer in [0,9]
```

Fig. 3.1. System grammar

A detailed function description can be found in (Tsakonas et al. 2004). In general, these functions can be divided into *topology altering functions* and *tuning functions*.

4. Results and discussion

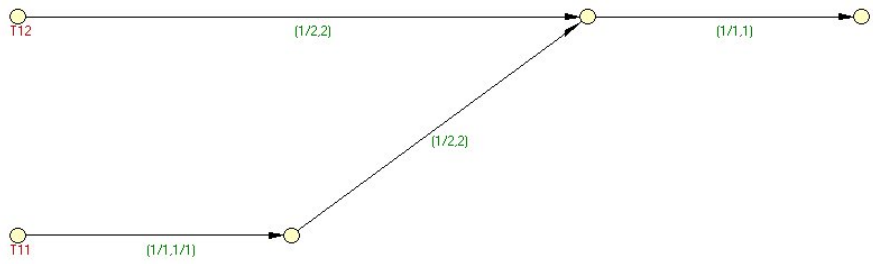
In the section that follows we apply the procedure in two real world problems, namely the shuttle-landing control problem and the ionosphere radar returns classification problem. Both databases reside in the *UCI Machine Learning* repository (Blake et al. 2003). In the first problem, the space shuttle landing control, we apply the fuzzy neural logic network model to achieve knowledge extraction. This problem is consisted by a very small database of NASA, which was used by the team of Roger Burke (Michie 1988) in order to determine the cases in which the crew of the space shuttle should prefer automated landing control, or perform the landing manually.. The features of the database are presented in *Table 3*, and they are comprised of weather conditions, landing error, and the shuttle stability. For each of the nominal input features having *n* possible values, we created *n* binary features. The solution shown in *Fig. 4.1* was achieved after 34,889 iterations. It achieved accuracy of 100% (5/5) in the test set (unknown data). The accuracy in the training set was 100% (10/10) and the accuracy in the validation set was also 100% (5/5).

Table 2. Shuttle landing data description

Domain	Selection of Automated or Manual Shuttle Landing procedure
Database	Shuttle-landing-control
Total Inputs	6 encoded to 12
Continuous Inputs	0
Discrete Inputs	2 encoded to 8 binary inputs
Binary Inputs	4
Total Records	15
Training Set Records	10
Validation Set Records	5
Test Set Records	5

Table 3. Features and their encoding of the Shuttle landing problem

Variable	Num	Feature	Values / Value range	Encoding
T1	#1	STABILITY	stab, xstab	1 (binary)
T2-T5	#2	ERROR	XL, LX, MM, SS	4 (nominal, 1 of 4)
T6	#6	SIGN	pp, nn	1 (binary)
T7	#7	WIND	head,tail	1 (binary)
T8-T11	#8	MAGNITUDE	Low, Medium, Strong, OutOfRange	4 (nominal, 1 of 4)
T12	#12	VISIBILITY	yes, no	1 (binary)



```
(CNLN (P1 (In T12) (S1 (In T11) (Link 78 0 (Link 35 0 (Link 88 0
(Rule 8 3)))) E)) (Link 2 6 (Rule 6 3)))
```

Fig. 4.1 Neural logic network for the Shuttle landing problem.

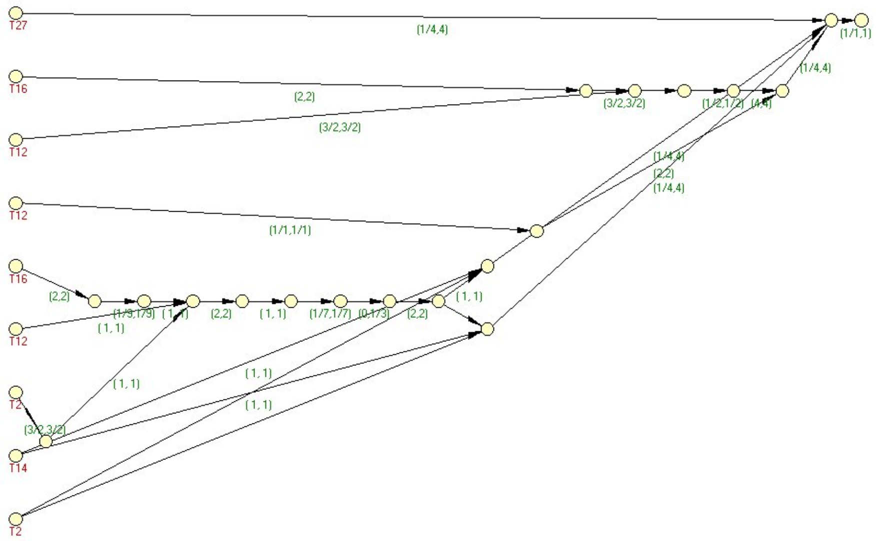
Hence, we may conclude that the extracted solution performs a perfect fit to the data. When we translate this network solution into a set of expert rules, we get the following expert system:

Q1 $\Leftarrow$ **Unanimity (**Magnitude Out of Range)
Q $\Leftarrow$ **Majority Influence (**Visibility, **Q1**)

This example demonstrates the ability of the system to provide potential knowledge discovery performed by the proposed methodology. The next domain of application for the neural logic network model is the ionosphere data returns classification.

Table 4. Ionosphere Radar data description

Domain	Classification of Ionosphere signals
Database	Ionosphere
Total Inputs	34
Continuous Inputs	34
Discrete Inputs	0
Binary Inputs	0
Total Records	346
Training Set Records	147
Validation Set Records	48
Test Set Records	151



```
CNLN (P1 (P1 (In T27) (S1 (P1 (S1 (S1 (S1 (In T14) (Rule 6 4) (S2 E
(Rule 4 4) E)) (Link 89 7 (Rule 4 6)) E) (Link 86 3 (Rule 11 2)) E)
(In T12)) (Rule 2 12) (P2 E (Rule 8 7) E))) (S1 (P1 (S1 (P1 (S1 (In
T16) (Rule 2 6) (S2 E (Rule 4 7) E)) (In T12)) (Link 196 2 (Link 85 3
(Link 193 3 (Rule 2 2)))) (S2 E (Rule 5 11) E)) (P1 (In T14) (In
T2))) (Rule 5 3) (P2 E (Rule 11 4) E))) (Link 2 2 (Link 191 2 (Rule 3
3))))
```

Fig. 4.3 Neural logic network for the Ionosphere Radar data classification problem

We selected the last 151 records as test set, according to the problem specifications in (Sigillito et al. 1989). The solution shown in *Fig. 4.3* was achieved after 50,000 iterations. It achieved accuracy of 92,05% (139/151) in the test set (unknown data). Other applications on this domain include C4 algorithms (94.0% accuracy), ID3 (96.7%), linear perceptron (90%), non-linear perceptron (92%) and backpropagation neural networks (96%) (Sigillito et al. 1989).

5. Conclusions and further research

This work presented a novel approach for the construction of neural logic networks. The proposed system inherits recent developments in genetic programming. It makes use of grammar guided search methodology and the cellular encoding advance in order to express arbitrary large and connected neural logic networks. The resulted solutions maintain their interpretability and they can be used either for knowledge discovery or as highly accurate classification systems. The effectiveness of the system is demonstrated in two real-world problems. The first problem concerns the shuttle landing control. In this problem, the system is shown capable of generating transparent solutions, by producing expert rules that enhance the domain knowledge. The next addressed problem is the ionosphere radar returns classification. The system is shown capable of producing competitive results, as compared to those of the existing literature. Further research will be applied in both domains, in order to explore extensively the solution space and possibly provide more simple and interpretable results. More data domains will be used in order to demonstrate the effectiveness of the proposed approach. In the part of the implementation, more logical operations will be included in future versions of the system. Finally, tuning the overall genetic programming parameters will be considered in order to offer the most proper search speed of the algorithm.

References

Blake C., Keogh E., Merz C.J., (2003) UCI Repository of machine learning databases. Dept. Inform. Comp. Sci. Univ. California, Irvine, CA. [http://www.ics.uci.edu/~mlearn/ML-Repository.html]

Chia H.W-K., Tan C-L.,(2001) Neural logic network learning using genetic programming, Intl.Journal of Comp.Intelligence and Applications, 1:4, 2001, pp 357-368

Gruau F. (1996) Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm", Ph.D. Thesis, Ecole Normale Superieure de Lyon, anonymous ftp:lip.ens-lyon.fr (140.77.1.11) pub/Rapports/PhD PhD94-01-E.ps.Z

Gruau F. (1996) On Using Syntactic Constraints with Genetic Programming, in P.J.Angeline, K.E.Jinnear,Jr., "Advances in Genetic Programming", MIT

Gruau F., Whitley D., Pyeatt L. (1996) "A Comparison between Cellular Encoding and Direct Encoding for Genetic Neural Networks", in Koza J.R., Goldberg D.E., Fogel D.B., Riolo R.L. (eds.), Genetic Programming 1996: Proceedings of the First Annual Conf., pp 81-89, Cambridge, MA, MIT Press

Janikow C.Z. (1996) A Methodology for Processing Problem Constraints in Genetic Programming, in Computers Math.Applic. Vol.32:8,pp 97-113

Koza J.R. (1992) "Genetic Programming: On the Programming of Computers by Means of Natural Selection", Cambridge, MA, MIT Press.

Koza J.R. (1999) Bennett III F.H., Andre D., Keane M.A., "Genetic Programming III", Morgan Kaufmann Publ., Inc.

Langley P., Simon H.A., Bradshaw G.L.(1983) Rediscovering chemistry with the Bacon system", in Machine Learning: an artificial intelligence approach, Vol 1, Morgan Kaufmann

Michie,D. (1988) The Fifth Generation's Unbridged Gap. In Rolf Herken (Ed.) The Universal Turing Machine: A Half-Century Survey, 466-489, Oxford University Press

Montana D.J. (1995) Strongly Typed Genetic Programming", in Evolutionary Computation, vol. .3, no. 2

Naur P.(1963) Revised report on the algorithmic language ALGOL 60, Commun. ACM, Vol 6, No 1, pp 1-17, Jan 1963

Prechelt L. (1994) Proben1 - A set of neural network benchmark problems and benchmarking rules", Tech.Rep. 21/94, Univ. Karlsruhe, Karlsruhe, Germany

Quah T-S., Tan C-L.(1995) Teh H-H.,Sriniivasan B.,"Utilizing a Neural Logic Expert system in Currency Option Trading", Expert Systems with Applications, 9:2, pp 213-222

Quah T-S., Tan C-L., Raman K., Sriniivasan B. (1996) Towards integrating rule-based expert systems and neural networks, Decision Support Systems, 17, pp 99-118

Ryan C., Collins J.J., O'Neil M. (1998) Grammatical Evolution: Evolving Programs for an Arbitrary Language, in W.Banzhaf, R.Poli, M.Schoenauer, T.C.Fogarty (Eds.), "Genetic Programming", Lecture Notes in Computer Science, Springer

Sfetsos A. (2000) A comparison of various forecasting techniques applied to mean hourly wind speed time series, Renewable Energy, 21, pp 23-25

Sigillito, V. G., Wing, S. P., Hutton, L. V.,Baker, K. B. (1989), Classification of radar returns from the ionosphere using neural networks. Johns Hopkins APL Technical Digest, 10, 262-266

Tan A-H., Teow L-N. (1997) Inductive neural logic network and the SCM algorithm, Neurocomputing 14, pp 157-176

Teh H-H. (1995) Neural Logic Networks, World Scientific

Tsakonas A., Aggelis V., Karkazis I., Dounias G. (2004) An Evolutionary System for Neural Logic Networks using Genetic Programming and Indirect Encoding, Journal of Applied Logic, Elsevier, forthcoming, Spring 2004