Evolving Neural-Symbolic Systems Guided by Adaptive Training Schemes: Applications in Finance

Athanasios Tsakonas [a] and Georgios Dounias [b] [1]

[a] Aristotle University of Thessaloniki,
Artificial Intelligence and Information Analysis Laboratory,
Dept. of Informatics, Thessaloniki, Greece

[b] University of the Aegean,
Dept. of Financial and Management Engineering,
31 Fostini Str., Chios, Greece

ABSTRACT: The paper presents a hybrid and adaptive intelligent methodology, based on neural logic networks and grammar-guided genetic programming. The aim of the study is to demonstrate how to generate efficient neural logic networks with the aid of genetic programming methods trained adaptively through an innovative scheme. The proposed adaptive training scheme of the genetic programming mechanism, leads to the generation of high diversity solutions and small sized individuals. The overall methodology is advantageous due to the adaptive training scheme proposed, for offering both, accurate and interpretable results in the form of expert rules. Moreover, a sensitivity analysis study is provided within the paper, comparing the performance of the proposed evolutionary neural logic networks methodology, with well-known competitive inductive machine learning approaches. Two financial domains of application have been selected to demonstrate the capabilities of the proposed methodology, (a) classification of credit applicants for consumer loans of a German bank and (b) the credit-scoring decision-making process in an Australian bank. Results seem encouraging since the proposed methodology outperforms a number of competitive existing statistical and intelligent methodologies, while it also produces handy decision rules, short in length and transparent in meaning and use.

KEYWORDS: adaptive training, symbolic connectionist systems, neural logic networks, grammar-guided genetic programming, hybrid and adaptive intelligence.

1. INTRODUCTION

Evolutionary neural logic networks (ENLN) consititute a recent advance in the domain of grammar-driven genetic programming (Tsakonas et al., 2004). The ENLN approach, introduced the production of neural logic networks and fuzzy neural logic networks which can be arbitrarily large and connected, and have the capability to maintain their interpretability. The neural logic networks (NLN) generally are powerful connectionist systems that have been applied in various

[1] Corresponding author. Tel: +302-271-094-408, Fax: +302-271-093-464
E-mail address: g.dounias@aegean.gr (Georgios Dounias)

domains (Teh, 1995), (Quah et al., 1995), (Quah et al., 1996), (Sfetsos, 2000) and by their definition can be easily interpreted in a number of expert rules. These networks can be considered as an integration between rule-based expert systems and neural networks (Quah et al., 1996). Although being powerful in their definition, the task of properly training a neural logic network has been problematic in the past. At first, 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 for the purpose of NLN-training. All 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 genetic programming (GP), (Chia and Tan, 2001). However, (Chia and Tan, 2001) in their system, provided a model capable of producing only a limited number of neural logic network representations, which resembled to a binary tree. In (Tsakonas et al., 2004), these problems have been adequately overcome. However, in order for this GP-guided NLN-algorithm to operate efficiently and effectively, several enhancements have been applied into the standard genetic programming training procedure.

This paper presents these advances, which enable the proposed ENLN-system to provide small and easily interpretable solutions, by controlling the *code-bloat* (Angeline 1998) using adaptive operation rates. In addition to the adaptive training scheme used, in the paper is also presented a sensitivity analysis of the proposed ENLN-system in comparison to a well-known inductive machine learning approach, the C4.5 algorithm (Quinlan, 1992). As a next step, the ENLN-system is applied in two problems belonging to the financial domain. The first problem corresponds to the credit applicants' classification for consumer loans of a German bank. The ENLN-based results obtained in this domain, demonstrate the ability of the proposed system to produce interpretable NLN-representations and facilitate the knowledge discovery process. The second selected financial application corresponds to a similar credit scoring decision problem of an Australian bank. Although feature details in the latter domain are not available, the application of the ENLN-system to this data enables conclusions to be drawn on its effectiveness in comparison to a number of statistical and intelligent approaches. Hence, the ENLN approach proved to be capable of investigating complex neural logic network structures with a high

classification rate, although it still maintains the ability of interpreting these networks into expert rules.

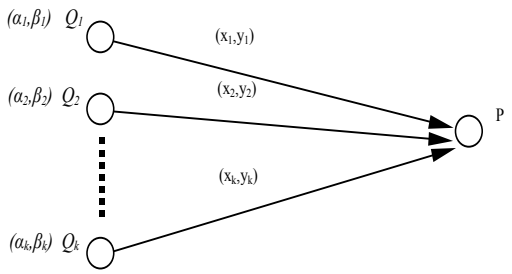
The paper is organised as follows. In section 2 a short presentation of the theoretical background of the neural logic networks and the genetic programming framework is made. Section 3 contains the system analysis of the ENLN methodology, with respect to the adaptive operation rates during training, the use of a validation set and the sensitivity analysis. In section 4, is given the application of the ENLN into two problems from the financial domain. Finally, section 5 contains conclusions regarding this work and proposes future research directions.

2. BACKGROUND

2.1 NEURAL LOGIC NETWORKS

A neural logic network can be represented as a finite directed graph. It usually consists of 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 an ordered pair weight (x, y) where x and y are real numbers. An example neural logic and its output value (a, b) of node P is shown in *Figure 1*.

Figure 1. The general form of a neural logic network and its output value



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

The rationale behind neural logic networks is to provide a connectionist system in which the following properties apply:

- The truth table of the output of a node corresponds to the truth table of a logical operation.
- Three-valued logic is supported (*true*, *false* and *don't know*).

- Any elementary network that corresponds to a basic logical operation may be combined with others to form larger networks that can perform complex logical decision tasks.
- Any such neural logic network should be interpretable into logical rules by simply interpreting its architecture and nodes.

In the example shown in *Figure 1*, we present the standard activation function for a neural logic node. As it can be seen, the output of such a node belongs to the set *{(1,0), (0,1), (0,0)}*. By using specific weights, different logical operations can be applied to the input nodes. Then, the result to the output node will be the same as defined in the truth table of the corresponding logical operation.

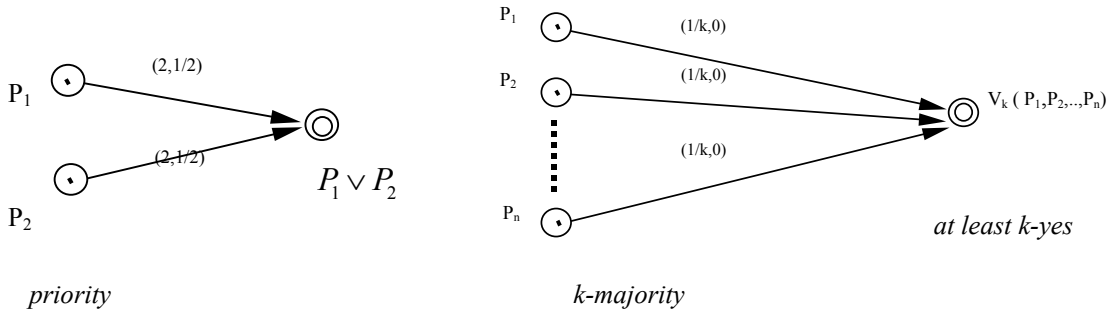


Figure 2. Example logical operators in neural logic networks

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. *Figure 2* depicts two such networks and their corresponding logical rule.

2.2 GENETIC PROGRAMMING

Currently, Genetic Programming (GP) has been applied in a wide range of real-world problems. GP-methodology, in its canonical form enables the automatic generation of mathematical expressions or, so-called, programs. Usually, a population of candidate solutions is maintained, and after a "generation" is completed, the population is expected to have higher fitness for a given

problem. The generic GP run algorithm can be summarized in the following steps (Koza et al., 2003):

1. Create an initial population at random consisted of individual computer programs.
2. Perform the following sub-steps until a termination criterion is satisfied:
    a. Using a fitness measure assign a fitness value to each individual.
    b. Create a new population by applying the three operators that follow. These operators are applied either to one or two individuals. Their selection is done using a tournament.
        i. Copy (reproduce) an individual without affecting it.
        ii. Create two new programs (offsets) by recombining sub-trees from two existing programs using the crossover operation at randomly selected tree points
        iii. Create a new program from an existing individual by mutating in one of the three following ways.

        iii-a. Mutate a node of the tree.

        iii-b. Mutate a number in a node of the tree.

        iii-c. Select a sub-tree and promote it to a higher node.

3.Extract the result (or the approximate result) and designate the best-so-far individual.

Having presented in short, the basic elements of the GP-approach, we move into an in-depth analysis of the proposed ENLN-system, presenting its adaptive characteristics and its sensitivity to noise and missing data.

3. ENLN SYSTEM ANALYSIS

The evolutionary approach that is used for the ENLN-system is composed of a grammar-driven, steady state genetic programming scheme. This system makes use of indirect encoding (cellular encoding) to represent arbitrarily sized and connected neural logic networks within the genetic programming trees. The population initialization and the overall training procedure are controlled by a grammar expressed in BNF notation. For further details, regarding the basic system architecture and functionality, the reader is strongly encouraged to address to (Tsakonas et al.,

2004). For the ENLN system, the population has been set to 2,000 individuals, a value that can be considered typical for GP (Koza, 1992) and is mainly related to hardware-memory constraint reasons. The selection of the individuals is accomplished using the tournament with elitist strategy (Koza, 1992), in which the individual to be selected is decided after a tournament among six members of a randomly selected population's subset, in respect to their fitness values. The selected size of members in the tournament (i.e. six) is also considered a typical value for the GP procedures. In order to perform a more effective search and also to avoid premature convergence, we choose to apply a *killing anti-tournament* (Koza 1992), in which two members of the population are selected randomly and the member with the lower fitness substitutes the other. This procedure aims to disable any possible domination of a highly fitted individual in the population. The maximum size of an individual is set to 650 nodes, which is a rather high value regarding common GP implementations, but permissible according to current hardware and memory capacities. The motivation for selecting a high maximum size of an individual is related to the requirements of the ENLN indirect encoding. Each run of the algorithm is applied at most up to 200 generations. All the abovementioned experimentation parameters are summarized in *Table I*.

Table I. GP training parameters

Parameter	Value
Population	2,000 individuals
Implementation	Grammar-driven steady state genetic programming
Selection	Tournament with elitist strategy
Tournament members	6
Killing anti-tournament	2
Maximum nodes allowed	650 nodes
Maximum generations number	200

3.1 ADAPTIVE TRAINING RATES FOR ENLN

The idea behind the adaptive training rate for the genetic operations is to ensure two things:

- To keep the diversity of the population in high levels
- To keep the average size of the population individuals low.

The first property provides a beneficiary framework for the algorithm during the solution search. The second property, confronts the program code bloat of the population, which often results in very large and non-interpretable solutions. Hence the target of this procedure can be clarified into two statements, which are eligible for the evolutionary neural logic networks:

- High classification rate
- Small (interpretable) solutions

In this regard, we examined three frameworks for the training process of the ENLN:

- Training with fixed rates for genetic operations (i.e. no adaptation at all)
- Training with adaptive rates based on threshold population values
- Training with adaptive rates directly related on the population size and growth

The first approach uses standard values for the genetic operations. These values have been extensively tested in our experiments hence we consider them near optimum, regarding a non-adaptive ENLN system. The values for these rates are shown in *Table II*.

Table II. Rate values for fixed rate operation of a ENLN system

Operation	Rate
Crossover	0.65
Node mutation	0.07
Constant mutation	0.07
Shrink mutation	0.21

Training using an adaptive rate based on threshold population values (called hereinafter as '*level*' adaptation), can be considered as an alternative to fixed rates for operations, the aim being here to apply a penalty to specific operators when a given level (threshold) of the average population size is exceeded. Within this framework, we apply the following three simple rules during the training procedure, see, Eqs. (1)-(3) below:

$$s_a + s_d > f_1 \cdot s_m \rightarrow r_c^1, r_m^1 \tag{1}$$

$$s_a + s_d > f_2 \cdot s_m \rightarrow r_c^2, r_m^2 \tag{2}$$

$$s_a + s_d < f_3 \cdot s_m \rightarrow r_c^3, r_m^3 \tag{3}$$

In the above equations, s_a is the average size of the individuals of the population, s_d is the standard deviation of the population individuals' size, s_m is the maximum allowed size of an individual, f_n is a constant for the *n*-case explained below, r_c^n is the crossover rate for the n-case and r_m^n is the total mutation rate for the *n*-case. These three rules maintain a different role in our system. The first rule (eq. (1)) reduces significantly the crossover operations vs. the mutation operations when the average size of the individuals exceeds a stated measure. The intention of the application of this rule is to beat the code bloat phenomenon that often arises after a number of operations. This rule takes precedence over rule of eq. (2). The second rule (eq. (2)) reduces mildly the crossover operations vs. the mutation operations when the average size of the individuals exceeds a stated measure. This is an auxiliary rule for rule of eq. (2), aiming to maintain the average size of the individuals in towardly levels. The third rule (eq. (3)) increases significantly the crossover operations vs. the mutation operations when the average size of the individuals goes below a stated measure. The intention of the application of this rule is to beat premature convergence that we may encounter in grammar-based systems if the population floods in the early stages of the algorithm with small, moderately good solutions. Individual values for the three mutation operators are obtained using a fixed analogy. More specifically, the following relations are used, (see Eqs. (4)-(6)):

$$ {}^{s}r_m^n = 0.6 \cdot r_m^n \tag{4}$$

$$ {}^{c}r_m^n = 0.2 \cdot r_m^n \tag{5}$$

$$ {}^{n}r_m^n = 0.2 \cdot r_m^n \tag{6}$$

In the above equations, ${}^{s}r_m^n$ is the *shrink mutation* (Singleton 1994) rate for the n-case, while ${}^{c}r_m^n$ and ${}^{n}r_m^n$ are the constant and node mutation rates accordingly. The constants f_n and the operation rates after experimentation take the values shown in *Table III*.

Table III. Constants values and operator rates for the *level* adaptive training

n	f_n	r_c^n	r_m^n
1	0.5	0.02	0.98

2	0.33	0.35	0.65
3	0.16	0.95	0.05

The next adaptive scheme tried on the ENLN is training with adaptive rates directly related on the population size and growth (called hereinafter as '*continuous'* adaptation). The rationale behind this approach is to maintain an online and imminent rate for the genetic operations, during the training phase, based on current population status and a momentum size factor. We have applied therefore the scheme described by the following equations (7)-(10):

$$r_m = f_1 \cdot \left(1 + \frac{s_t}{f_2} - f_2 \cdot s_e \right) \tag{7}$$

$$r_c = 1 - r_m \tag{8}$$

$$s_t = \frac{s_a^t - s_w}{s_w} \tag{9}$$

$$s_e = \frac{s_a^t - s_a^{t-1}}{s_a^{t-1}} \tag{10}$$

In the above equations, f_1 is the "starting value" mutation rate, set here to 0.7, f_2 is the "adjusting" factor, set here equal to 1.2 (a value of 1.0 corresponds to equal impact of the s_t and s_e parameters), s_e is the relative increment size (a momentum value), s_t is the relative trend size, s_w is the size where we prefer that the average population is being driven to (set here to 150 nodes), and finally, s_a^t and s_a^{t-1} are the average population sizes at the *t* and *t-1* generations. In this scheme, shrink mutation rate is additionally adapted within the total mutation rate, using the following relation given by eq. (11):

$$r_m^s = R_m^s \cdot \frac{s_a^t}{s_w} \tag{11}$$

In the above relation, R_m^s is the maximum allowed relative mutation rate for the shrink mutation operator (set here to 0.9), s_a^t is the average population size at the *t*-generation, r_m^s is the shrink mutation rate relative to the overall mutation rate. The above relations form a framework in which crossover and mutation rates are adapted after every generation. However, in order to perform more effectively, we form a limit to the overall mutation rate, disabling values higher

than 0.9, thus enabling always the existence of crossover in a sensible amount. Additionally, we apply the same constraints to the shrink mutation rate relative to the overall mutation rate (i.e. maximum allowed rate set to 0.9).

The results from the training process are illustrated in Figures 3, (a)-(h). In Figure 3(a) and (b), is shown the number of the performed crossover and mutations accordingly. One may easily observe the different operating behavior of these three approaches, and especially the high mutation rates adapted by the *level* and the *continuous* adapting scheme. Figure 3(c) depicts the shrink mutation operations, demonstrating the special handling of this operator in the continuous adapting training, in an attempt to control the population size. Figure 3(d) shows the average population size of the ENLN system during training. It is clearly shown that the adaptive approaches handle better the code bloat effect by reducing the average population size to the desirable levels. Between the two adaptive approaches, the continuous adaptation scheme succeeds in keeping the population in constant levels more effectively than the level adaptation. It is shown also that delays the increment of the code significantly during the early stage of training. In Figure 3(e), we present the standard deviation of the population size. This feature can be a measure of population diversity. In this respect, the continuous adaptation seems to perform poorer in the early stage of training, than the level adaptation and the no-adapt scheme. Figure 3(f) shows the average fitness of the population during training. Suggesting that best individual's fitness is equal, or nearly equal, high values of the average fitness may denote a low-diversity population. As it is shown in this figure, the no-adapting scheme has performed worst in this case, having the level-adaptation performing better after the early training stage. In Figure 3(g), we depict the standard

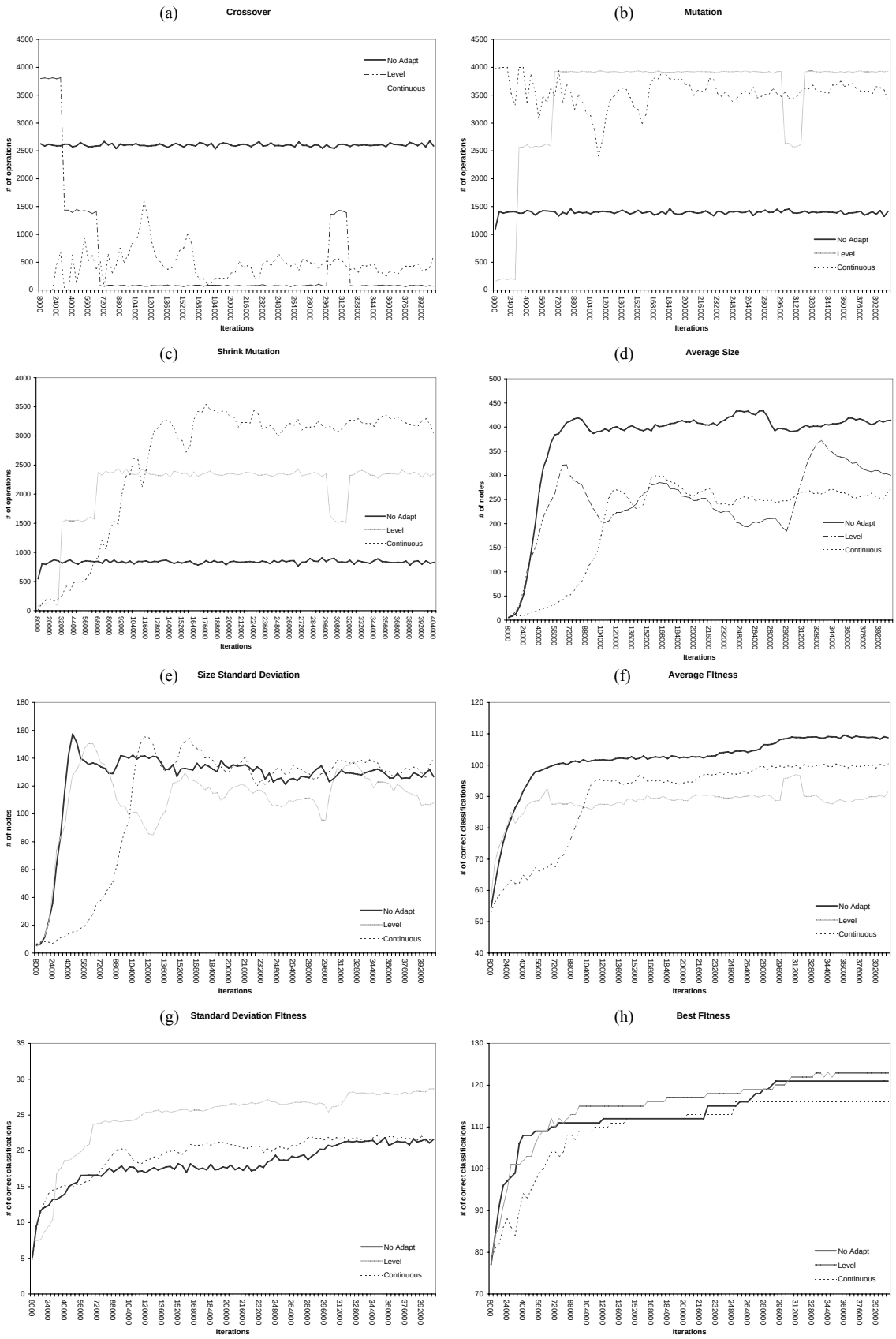


Figure 3. Performance of the training schemes: (a) crossover operations, (b) overall mutation operations, (c) shrink mutation operations, (d) average population size, (e) size std. deviation, (f) average fitness, (g) fitness std. deviation and (h) best fitness.

deviation of the population fitness. High values of standard deviation are positive sign of population diversity. In this respect, the level adaptation has performed better, with the no-adapt scheme having the lowest standard deviation. Finally, Figure 3 (h), shows the best individual's fitness during training process. As it is seen in this figure, the level adaptation prevails to the other two approaches, with the continuous adaptation having slightly lower values than those of the no-adapting approach. In respect to the above experimentations, we selected to apply the *level adaptation* throughout our work.

Having described the adaptive training schemes for the ENLN, in the next paragraph we continue with the methodology used in order to guide the search process and avoid overfitting to the training data set. The methodology selected to be applied, is the incorporation of a *validation set*.

3.2 VALIDATION SET

Currently, the most popular procedure in literature to avoid overfitting in the training set is the use of a validation set. This technique consists of the partition of the subset used for training into two parts. The first part is used for the main training of the algorithm, and the second part is used for validation. In this respect, as best solution is selected the one that maintains the lowest classification error in *both* the training and the validation set. More specifically, this procedure in ENLN is applied as follows.

Suppose $F_i^t, F_i^v, i = 1,..,n$, where F_i^t is the fitness value of the best individual in the training set after i generations, F_i^v is the fitness value of the best individual in the validation set after i generations and n is the maximum number of generations, we get the following equation (12):

$$V_{best} \Leftarrow V_i, iff \{(F_i^t > F_{best}^t) \wedge (F_i^v \geq F_{best}^v)\} \quad (12)$$

where V_{best} is the (final) best solution, V_i is the best solution after i generations, F_{best}^t is the fitness value of the (final) best individual in the training set and F_{best}^v s the fitness value of the (final) best individual in the validation set. In fact, aiming to obtain smaller size of the solutions we slightly modify the above equation (12) to the following one (13):

$$V_{best} \Leftarrow V_i, iff \begin{Bmatrix} (F_i^t > F_{best}^t) \wedge (F_i^v \geq F_{best}^v) \\ \vee \\ (F_i^t = F_{best}^t) \wedge (F_i^v \geq F_{best}^v) \wedge (K_i < K_{best}) \end{Bmatrix} \quad (13)$$

In the above equation (13), K_i is the size (in nodes) of the best solution after i generations, and K_{best} is the size (similarly in nodes) of the final best solution acquired.

3.3 SENSITIVITY ANALYSIS

In data mining tasks, an essential property for a system is the ability to learn by incomplete and imperfect data sets. The usual types of imperfections encountered are (Wong, 2001):

- Random noise to the data.
- Very small data size.
- The distribution of the training data, which fails to represent the underlying distribution of the data domain.
- The use of improper language description and the selection of wrong features.
- It is possible to fail to represent important features or to accentuate irrelevant data features. It is also possible that the language description cannot include an exact description of the data domain.
- The existence of missing values in the training set.

Existing inductive learning systems apply various techniques to manage noise in order to encounter the first five imperfections above. The management of the missing values is usually performed in a different manner. The noise management techniques have been designed in such way in order to prevent the overfitting of the system to the imperfect training data set, by excluding the non-important elements of the data (Lavrac and Dzeroski, 1994). Techniques of this kind are the pruning of the decision trees in CART, the rule cutting in AQ15 (Leung and Wong, 1991) and the meaningfulness checking in CN2 (Clark and Niblett, 1989). However, these techniques may ignore some important features of the data domain, just because they are not of statistical importance. Moreover, the above learning systems use a limited feature-value language to represent the training data sets and the connoted knowledge. Today, only a limited number of

learning systems like FOIL (Quinlan 1990), (Quinlan, 1991), mFOIL (Lavrac and Dzeroski, 1994) and C4.5 (Quinlan, 1992) handle the task of learning by imperfect data. In order to obtain an analytic description of the ENLN system behaviour under different environment conditions, we investigate its performance in comparison to a well-known computational intelligence algorithm. More specifically, we examine the system's behaviour with respect to the following attributes:

- Ability to learn from noisy data
- Ability to learn from data with missing values
- Ability to perform feature selection

The algorithm we have selected to compare to, is the well-known C4.5, belonging to inductive machine-learning, which produces inductive decision trees (Quinlan, 1992). The methodology of the inductive decision trees uses as a key-measure the information entropy. C4.5 is a very popular methodology and one of the most frequently applied intelligent systems in data mining tasks. The problem addressed is composed by synthetic data artificially constructed for this reason. For the generation of the data sets, we applied the DataGenerator (DataGen, 2004), a system that uses rules to produce random records. The data produced by DataGenerator are mainly addressed by rule-building systems (like the C4.5 and the ENLN), and they can be used to obtain comparative values of their performance. For the creation of the data, it is used a rule set in conjunctive and disjunctive form, with a complexity degree defined by the user. It is possible to select the embedment of different noise levels and/or different number of missing values in the data set. We selected to create a data set consisting of 300 records, from which the first 200 are used for the systems' training phase and the rest 100 as testing data set (unknown data). For the conducted experiments, we used a validation set consisting of the last 20 records of the 200-record training data set. Summarizing, the proposed ENLN-system in each experiment used the first 180 records as training set, the next 20 records as validation set and the last 100 as a test set. The C4.5 system used the first 200 records as training set and the last 100 as test set[2].

Two classes compose the examined domain. We set the distribution of these classes to be uniformly random. We also decided to include four features (*A*, *B*, *C* and *D*), from which one (D) will not participate in the rules that define the classes, aiming to check the ability of ENLN to

[2] The data used and the analytic results are available to download from: http://decision.fme.aegean.gr/analysis/

perform feature selection, in which the presence of the D in every solution is undesirable. The problem is composed by the following two rules:

- If A=[5,6] and B=[5,6] then *c1* (53.0%)
- If A=[9,10] and B=[5,6] and C=[8,9] then *c2* (47.0%)

We created the following data sets using the aforementioned properties. A 'noisy' record is consisted here of a partially false predicate (i.e. *If A=9 and B=6 then c1*) whereas a 'missing' value record is a record with partially unknown predicate (i.e. *If A=? and B=6 then c1*).

- Sets with noise 0%, 5%, 10%, 20%, 33%, 50%, 80% and 100%
- Sets with missing values 0%, 5%, 10%, 20%, 33%, 50% and 80%

We performed the training procedure for each algorithm once for every data set. In *Table IV* we present the classification error in data sets with noise.

Table IV. Classification error in noisy data sets

Data noise	% Classification error of ENLN in test set	% Classification error of C4.5 in test set
0%	0	0
5%	3	3
10%	3	5
20%	6	4
33%	13	11
50%	14	29
80%	36	47
100%	38	41

In this table we may observe that the ENLN system performs better in 5 out of the 7 cases than C4.5. Specifically C4.5 is shown to maintain lower error rates in the "middle" noise levels (20% and 33%), while the ENLN system succeeds in low and high noise levels. These results are shown in *Figure 4.* It is worth to note that the success of C4.5 can be modified only if we modify the algorithm parameters; on the contrary, by performing repetitive runs, and still maintaining the same ENLN system setup, there is a potential of achieving better results, which is due to ENLN 's stochastic nature.

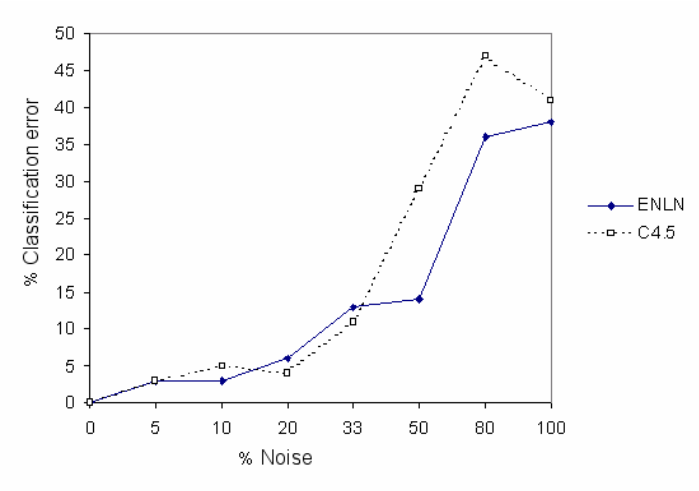


Figure 4. Classification error in noisy data for ENLN and C4.5

The next experimentation was performed in a data set with missing feature values. The results are presented in *Table V*.

Table V. Classification error rate in data sets with missing feature values.

Missing values percentage	% Classification error of ENLN in test set	% Classification error of C4.5 in test set
0%	0	0
5%	3	4
10%	3	6
20%	7	18
33%	17	11
50%	25	32
80%	40	43

In this table we may observe that the proposed ENLN-system outperforms the C4.5 with the exception of the case where the missing values comprise 33% of the total. In 5 out of the 6 cases the ENLN approach achieved lower error rates than those of the C4.5. These results are shown in *Figure 5*.

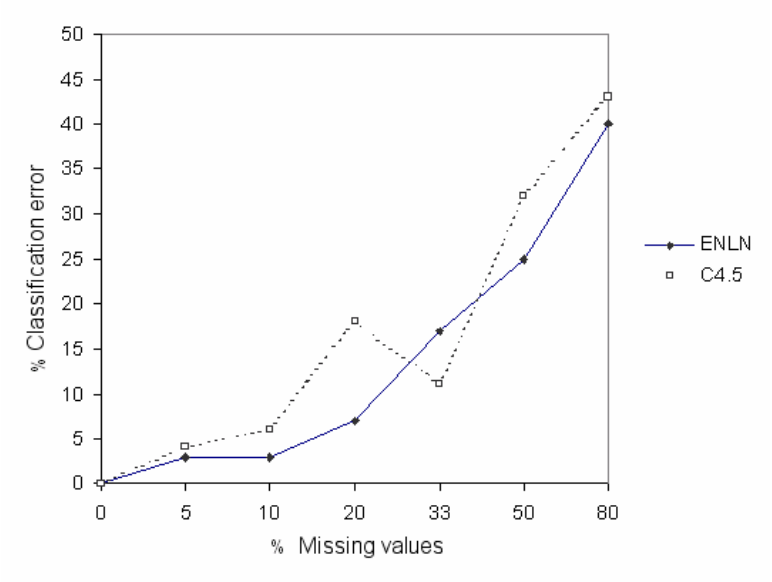


Figure 5. Classification error rate in data sets with missing values for ENLN and C4.5

By definition, the ENLN system during the training phase tries to select only those features, which are considered to participate in the data domain. Hence, the user is not bound to pre-select a number of the features as the system inputs, and moreover, after the training process the user may extract useful conclusions based on the inclusion or the rejection of specific features. This property is known as feature selection, and is usually a characteristic met in machine learning systems. More specifically, the C4.5 algorithm is also used for feature selection. As previously shown, we entered to our experimental datasets a variable which does not participate to the classes definitions, in order to investigate the performance of ENLN in feature selection. The results are shown in *Table VI*. As it can be seen, the undesirable variable is present in the ENLN solution in 8 out of 14 cases. Respectively, this undesirable variable is present in the C4.5 solution in 7 out of the 14 cases. Six of the cases that involve the undesirable variable are common to both systems. Hence, the performance of the ENLN-system should be considered a highly competitive one, since systems such as C4.5 are widely used for years as smart feature selection tools.

Table VI. Inclusion of undesirable variable in ENLN and C4.5 systems

Data set	Undesirable variable included	Undesirable variable included

	in the ENLN solution	in the C4.5 solution
0% noise	No	No
5% noise	Yes	No
10% noise	Yes	No
20% noise	No	No
33% noise	No	No
50% noise	No	No
80% noise	Yes	Yes
100% noise	Yes	Yes
5% missing values	Yes	Yes
10% missing values	Yes	Yes
20% missing values	Yes	Yes
33% missing values	No	No
50% missing values	No	Yes
80% missing values	Yes	Yes

As a concluding remark, the ENLN system, compared to the C4.5 algorithm proved:

- Better in handling noisy data in 5 out of the 7 of the examined cases.
- Better in handling data with missing values in 5 out of the 6 of the examined cases.
- Competitive to C4.5, with respect to its ability to perform feature selection tasks.

Having examined the algorithm in terms of its sensitivity to noise and missing data, we now proceed in presenting the application of the system into two problems from the area of financial decision-making.

4. APPLICATIONS IN FINANCE: RESULTS AND DISCUSSION

4.1 GERMAN BANK CREDIT SCORING PROBLEM

In this section we proceed in testing the proposed ENLN algorithmic approach, into a real-world classification problem from the financial domain. The problem addressed, is related with the creditability of the applicants of a German bank for consumer loans (Fahrmeir and Tutz, 1994). Available features are financial and personal properties of the applicant, like real estate, sex etc. Data consist of both, continuous and discrete features, and there are no missing values. We used the encoding proposed by the data base creators (Fahrmeir and Tutz 1994) since a large number of these features are already encoded properly for the system input variables. *Table VII* shows details of the domain data, while the corresponding attribute descriptions are presented in *Table VIII*. We have adopted the encoding scheme use by the data base creators. It is worth to note however, that this encoding includes a number of problematic cases, in which variables with discrete

independent values are encoded into non- independent values. Such case is for example the value T7. As a consequence, the results in all algorithms are expected poorer than the potential.

Table VII. Data domain description for the German credit-scoring problem

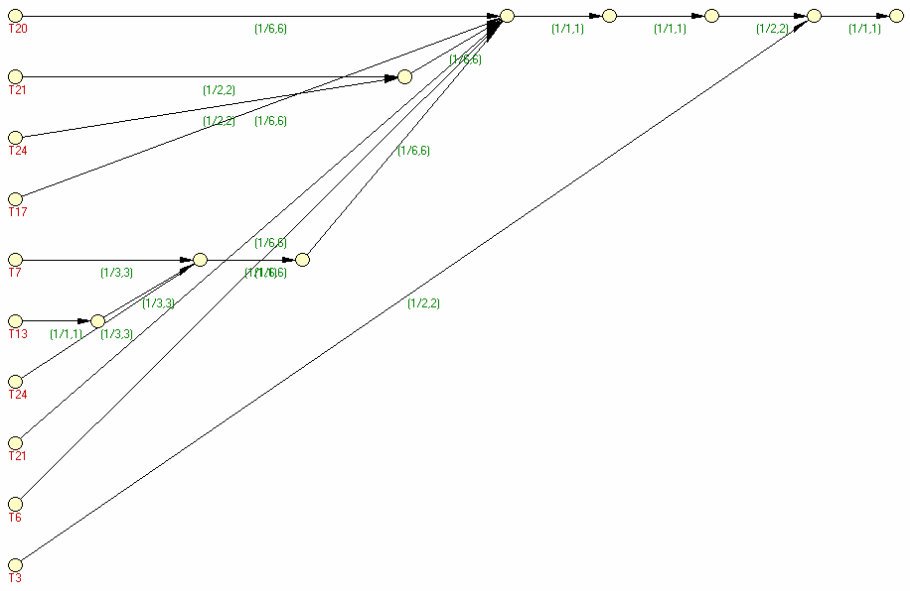
Domain	German credit scoring problem
Database name	German
Input features	20
Continuous input features	2
Discrete input features	18
Binary input features	0
Anti – Overfitting method	Validation set
Total records	996
Training set records	498
Validation set records	249
Test set records	249
Missing data	No
Data standardization/normalization	No

Table VIII. Feature description for the German credit-scoring problem

Variable	Feature	Values / Value range	Encoded features
T1	Current account status (salaries for at least one year) , in thousands	2: x < 0 DM, 3: 0 ≤ x < 200 DM, 4: x ≥ 200 DM, 1: no account	1 (discrete) → 0...1
T2	Month duration	Continuous	1 (continuous) → 0...1
T 3	Credit history	2: no credits / all credits paid in time 4: all credits in this bank paid in time 3: existent credits paid in time up to now 0: delay in the past to pay 1: problematic account / there are other credits (not in this bank)	1 (discrete) → 0...1
T 4	Credit size	Continuous	1 (continuous) → 0...1
T 5	Savings account / income bonds, in thousands	2: x < 100 DM, 3: 100 ≤ x < 500 DM, 4: 500 ≤ x < 1000 DM, 5: x ≥ 1000 DM, 1: no account / unknown account	1 (discrete) → 0...1
T 6	Current occupation	1: unemployed, 2: x < 1 year, 3: 1 year ≤ x < 4 years, 4: 4 years ≤ x < 7 years, 5: x ≥ 7 years,	1 (discrete) → 0...1
T 7	Personal status and sex	1: male, married 2: female, divorced / widowed / married 2: male, unmarried 3: male, divorced / widowed 4: female, unmarried	1 (discrete) → 0...1
T 8	Current home	1: x < 1 year, 2: 1 year ≤ x < 4 years, 3: 4 years ≤ x < 7 years, 4: x ≥ 7 years,	1 (discrete) → 0...1
T 9	Property	4: real estate 3: if without real estate: savings contract of construction company / life insurance 2: if without real estate: car 1: without real estate / unknown	1 (discrete) → 0...1
T 10	Age in years	1: x ≤ 25 years, 2: 26 years ≤ x ≤ 39 years, 3: 40 years ≤ x ≤ 59 years, 5: 60 years ≤ x ≤ 64 years, 4: x ≥ 65 years,	1 (discrete) → 0...1
T 11	Other accounts	0: in other bank 1: in other branches 2: none	1 (discrete) → 0...1

Variable	Feature	Values / Value range	Encoded features
T 12	Number of existing accounts in this bank (including current)	1: one 2: two or three 3: four or five 4: six or more	1 (discrete) → 0...1
T 13	Number of people being liable to provide maintenance	2: from 0 to 2 1: 3 or more	1 (discrete) → 0...1
T 14	Telephone	1: no 2: yes, at the name of potential customer	1 (binary) → 0...1
T 15	Foreign worker	1: yes 2: no	1 (binary) → 0...1
T 16- T 18	Reason for loan	0: new car purchase 1: used car purchase 2: furniture purchase 3: TV/radio purchase 4: local use 5: repairs 6: education 7: holidays 8: reeducation 9: business reasons 10: other reasons	1 (discrete) → 3 binaries 0-1
T 19-T 20	Other guarantees	0: none 1: co-applicant 2: guarantee	1 (discrete) → 2 binaries 0-1
T 21- T 22	Residence	2: hire 3: privately owned 1: residence for free	1 (discrete) → 2 binaries 0-1
T 23- T 24	Work	0: unemployed – non permanent 1: uneducated permanent 2: skilled employee / officer 3: management/ self-employed / high ranking employee	1 (discrete) → 2 binaries 0-1

The solution achieved is shown in Figure 6. This network classified successfully 72.69% (181/249) of the test data set (i.e. unknown data). The classification score in the training and validation set reached 72.69% (362/498) and 69.1% (172/149) respectively.



(CNLN (P1 (S1 (P1 (P1 (P1 (In T20) (P1 (S1 (P1 (In T21) (In T24)) (Rule 0 0) E) (In T17))) (P1 (S1 (P1 (P1 (In T7) (S1 (In T13) (Link 214 0 (Rule 0 0)) E)) (In T24)) (Link 106 0 (Rule 0 0)) (S2 E (Rule 0 0) E)) (In T21))) (In T6)) (Rule 0 0) (S2 (S2 E (Rule 0 0) E) (Rule 0 0) E)) (In T3)) (Rule 0 0))

Figure 6. Neural logic network (description and representation) created for the German credit-scoring problem.

The above network corresponds to the following set of rules:

- Q1 ⇐ **Conjunction (□□D) (**personal status and sex, Number of people being liable to provide maintenance for, Skilled employee/officer)
- Q2 ⇐ **Conjunction (□□D) (**Own housing, Skilled employee/officer)
- Q3 ⇐ **Conjunction (□□D) (Q1, Q2**)
- Q ⇐ **Conjunction (□□D) (Q3**, Credit history)

In the abovementioned rule set we may draw the following further conclusions:

- The credit history has significant importance for the decision-making process, since its value is applied in a conjunctive rule with the result of all previous logical operations from the rest of the features used.
- Another important feature, for the bank's decision-making task, seems to be the professional status of the applicant. Its value is applied in two logical operations (Q1 and Q2)
- Only 5 out of 24 available features were selected in total to be applied by the ENLN-system
- Complex decision rules seem not to be necessary, since the produced set of decision rules, uses only conjunctions (i.e. "AND" operations).

4.2 AUSTRALIAN BANK CREDIT SCORING PROBLEM

In the second financial application selected to demonstrate the effectiveness of the proposed ENLN-system, we investigate a similar with the previous credit applicants' evaluation problem, this time using a set of data acquired by an Australian bank (Quinlan, 1992). The features in this data are given as simple elements and their interpretation is not known, since this data set has been considered as classified information. However, the application of the ENLN system to this decision-making problem may provide useful conclusions regarding the system's effectiveness, since there exist enough successful applications of other approaches (Quinlan, 1987), (Statlog, 2002), in related literature. *Table IX* provides the description of the domain, while *Table X* presents the related attributes and the encoding that was used.

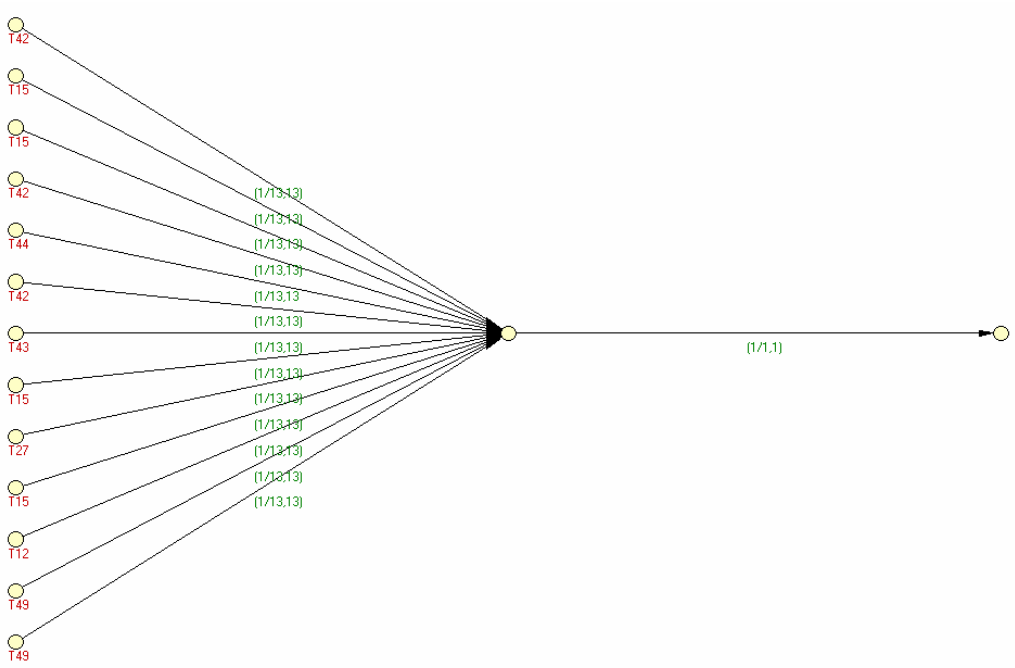
Table IX. Data domain description for the Australian bank credit-scoring problem

Domain	Australian bank credit scoring problem
Database name	Australian / CRX
Input features	14 encoded to 51
Continuous input features	6 encoded to 6
Binary input features	3
Discrete input features	6 encoded to 42
Anti – Overfitting method	Validation set
Total records	688
Training set records	344
Validation set records	172
Test set records	172
Missing data	Yes, 67 values
Data standardization/normalization	No

This data is primarily composed by discrete attributes, which are split into independent binary features for further processing by our system. The percentage of cases for which the application was finally accepted, reaches 44.5 % of the total (307 records). A total of 37 records have one or more missing values (5% of total data). After the training process, the ENLN-algorithm finally generated the specific neural logic network shown in *Figure 7*. The classification rate of this solution in the test set (unknown data) reaches 89.53% (154/172) which is higher than those reported in literature (Quinlan, 1992). The corresponding classification rates in the training and the validation set were 85.47% (294/344) and 87.21% (150/172) respectively.

Table X. Feature description for the Australian bank credit-scoring problem

Variable	**Values / value range**	**Encoded features**
T1-T3	discrete (b, a, ?), 12 missing values	1 (discrete) → 3 binaries, 1 of 3 (b="1 0 0", a="0 1 0", ?="0 0 1")
T4-T5	continuous (13.75...80.25)	1 (continuous) → 2 (1 continuous 0..1, 1 binary, 1: absent 0: non-absent)
T6	continuous (0...28)	1 (continuous) → 0..1
T7-T11	discrete (u, y, l, t, ?), 6 missing values	1 (discrete) → 5 binaries, 1 of 5
T12-T15	discrete (g, p, gg, ?), 6 missing values	1 (discrete) → 4 binaries, 1 of 4
T16-T30	discrete (c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff, ?), 9 missing values	1 (discrete) → 15 binaries, 1 of 15
T31-T40	discrete (v, h, bb, j, n, z, dd, ff, o, ?), 9 missing values	1 (discrete) → 10 binaries, 1 of 10
T41	continuous (0...28.5)	1 (continuous) → 0..1
T42	Binary	1 (binary) → 0..1
T43	Binary	1 (binary) → 0..1
T44	continuous (0...67)	1 (continuous) → 0..1
T45	Binary	1 (binary) → 0..1
T46-T48	discrete (g, p, s)	1 (discrete) → 3 binaries, 1 of 3
T49-T50	continuous (0...2000), 13 missing values	1 (continuous) → 2 (1 continuous 0..1, 1 binary, 1: absent 0: non-absent)
T51	continuous (0...10000), 13 missing values	1 (continuous) → 0..1



(CNLN (P1 (P1 (In T42) (P1 (In T15) (P1 (In T15) (P1 (P1 (In T42) (P1 (In T44) (P1 (P1 (P1 (In T42) (P1 (In T43) (P1 (In T15) (In T27)))) (In T15)) (In T12)))) (In T49))))) (In T49)) (Rule 0 0))))

Figure 7. Neural logic network (description and representation) generated for the Australian credit-scoring problem

The generated neural logic network can be described by the following extremely simple decision rule:

Q ⇐ **Conjunction (AND) (**T12, T15, T15, T15, T15, T27, T42, T42, T42, T43, T44, T49, T49**)**

The main conclusions regarding this extracted decision-rule follow:

- Only simple attribute conjunctions (AND operations) are contained in the resulting NLN, a fact showing that no complex decision-rules should be expected for building up an effective decision –making strategy from the bank's viewpoint, regarding credit applicants' evaluation.
- Getting into greater detail, attribute (feature) T15 seems to be of significant importance for the overall decision-making process. This feature is binary. The initial data feature from which T15 was derived, receives values from the set {g, p, gg, ?}, with T15 corresponding to the last value.
- Feature T42 is also of significant importance for the decision-making process.
- Another significant feature is the T49, which in the initial data set receives values within the range [0,2000].

The Australian bank data problem is particularly interesting, since it enables the comparison of the ENLN with a number of competitive statistical and intelligent approaches. Table XI presents the ENLN-results as compared with 22 other competitive approaches. Among the provided methodologies of Table XI, the ENLN-approach succeeded to obtain the highest classification score using the specific neural logic network presented above (Fig. 7).

Table XI. Classification rates in unknown data for the Australian credit-scoring problem. Results from (Statlog 2002).

Methodology	**Classification rate in unknown data (%)**
Cal5	86.9
Itrule	86.3
LogDisc	85.9
Discrim	85.9
Dipol92	85.9
Radial	85.5
Cart	85.5
Castle	85.2
Bayes	84.9
IndCart	84.8
BackProp	84.6
C4.5	84.5
Smart	84.2
BayTree	82.9
KNN	81.9
Ac2	81.9
NewId	81.9
LVQ	80.3
Alloc80	79.9
Cn2	79.6
QuaDisc	79.3
Default	56.0
NLN	**89.5**

5. CONCLUSIONS AND FURTHER RESEARCH

The paper discussed the effectiveness of an innovative hybrid and adaptive intelligent methodology, based on neural logic networks and grammar-guided genetic programming. Initially the proposed overall methodology was discussed, for generating efficient neural logic networks with the aid of genetic programming methods trained adaptively through an innovative scheme. Then description and discussion of a proposed novel adaptive training scheme of the GP-process followed, which successfully leads to the generation of high diversity solutions and small sized individuals, in other words to the achievement of highly accurate, short in length and easily interpretable results, having the form of logical expert rules. Then, a sensitivity analysis study was provided, for comparing the performance of the proposed ENLN-methodology, with well-known

competitive inductive machine learning approaches (C4.5 was selected to be the tool for comparison). The comparison was made on artificially produced datasets with varying conditions of embedded noise, missing values and insertion of dummy variables for checking the ability for feature selection. Results were encouraging according to the comparison performed, with the ENLN-system outperforming C4.5 in noisy conditions and missing values and performing comparatively in feature selection tasks.

Finally, two financial domains of application were selected to demonstrate the capabilities of the proposed methodology:

(a) Classification of credit applicants for consumer loans of a German bank

(b) A credit-scoring decision-making process concerning an Australian bank.

Results were encouraging as well, for both domains, since the proposed ENLN-methodology achieved the highest classification accuracy among, a large number of competitive existing statistical and intelligent methodologies, while it also generated a rather small set of particularly meaningful and handy decision rules for the management of the credit applicants' decision-making problem.

The authors currently extend their research work towards a number of open problems regarding the proposed ENLN methodology, such as:

- the attempt to construct evolutionary recursive high-order neural networks with interpretable outcome,
- further investigation on the adaptive training methodologies within the ENLN framework
- the application of the ENLN-approach in specific multi-class and multivariate time-series forecasting problems,
- the incorporation of the minimum description length principle or other metrics into the training procedure.

Additionally, as shown in *Table III*, after experiments, we concluded that these parameters are optimal for our ENLN system setup. Hence, these values are heuristically obtained. We believe that it would be worthwhile however, in a future work to obtain a relationship between all major ENLN system parameters (including grammar size, population size, maximum allowed individual size etc.) and these tuning values.

.

6. REFERENCES

Angeline P.J., Subtree crossover causes bloat, Genetic Programming 1998: Proceedings of the Third Annual Conference, J.R.Koza, W.Banzhaf, K.Chellapilla, K.Deb, M.Dorigo, D.B.Fogel, M.H. Garzon, D.E.Goldberg, H.Iba, R.Riolo (Eds.), pp. 745-752, University of Winsconsin, Winsconsin, 1998, Morgan Kaufmann

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

Clark, P., Niblett, T., 1989. The CN2 induction algorithm, Machine Learning, 3: 261-283

DataGen. 2004. DataSet Generator, http://www.datasetgenerator.com/overview.html

Fahrmeir, L., Tutz, G. 1994. Multivariate Statistical Modeling Based on Generalized Linear Models. New York: Springer.

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

Koza, J., Bennett, F., Andre, D., Keane, M. 2003. Genetic Programming III: Automatic Programming and Automatic Circuit Synthesis. Morgan Kaufmann.

Lavrac, N., Dzeroski, S. 1994. Inductive Logic Programming: Techniques and Applications. Ellis Horwood Series in Artificial Intelligence. Ellis Horwood, Chichester.

Leung, K.S., Wong, M.L. 1991. Automatic refinement of knowledge bases with fuzzy rules, Knowledge-Based Systems, 4, pp 231-246.

Quah, T-S., Tan, C-L., Teh, H-H., Sriniivasan, B. 1995. 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.

Quinlan, J.R. 1987. Simplifying decision trees. International Journal of Man-Machine Studies 27, December: pp. 221-234.

Quinlan, J.R. 1990. Learning logical definitions from relations. Machine Learning, 5, pp. 239-266.

Quinlan, J.R. 1991. Determinate literals in inductive logic programming, Proceedings of the 8th International Workshop on Machine Learning. pp 442-446, California, USA: Morgan Kaufmann.

Quinlan J.R. 1992, C4.5: Programs for Machine Learning, Morgan Kaufmann.

Singleton A., 1994, Genetic Programming with C++, BYTE Magazine, February 1994

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

Statlog 2002. Statlog Use, Test of Australian Credit Scoring data, (http://www.liacc.up.pt/ML/statlog/datasets/australian/australian.use.html)

Tan, A-H., Teow, L-N. 1997. Inductive neural logic network and the SCM algorithm. Neurocomputing: Vol. 14, 2 : 5, pp.157-176.

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

Tsakonas, A., Aggelis, V., Karkazis, I., and Dounias, G. 2004. An Evolutionary System for Neural Logic Networks using Genetic Programming and Indirect Encoding. Journal of Applied Logic: 2 (3), pp. 349-379.

Wong, M.L., 2001. A flexible knowledge discovery system using genetic programming and logic grammars. Decision Support Systems, 31: pp. 405-428.