Symbolic-Neural Rule Based Reasoning and Explanation

Ioannis Hatzilygeroudis[a], Jim Prentzas[b*]

[a] University of Patras, School of Engineering
Department of Computer Engineering & Informatics,
26500 Patras, Greece
+30-2610-996937
ihatz@ceid.upatras.gr

[b] Democritus University of Thrace, School of Education Sciences
Department of Education Sciences in Early Childhood, Laboratory of Informatics
68100 Nea Chili, Alexandroupolis, Greece
+30-25510-30086
dprentza@psed.duth.gr
*alphabetic order

Abstract

In this paper, we present neurule-based inference and explanation mechanisms. A neurule is a kind of integrated rule, which integrates a symbolic rule with neurocomputing: each neurule is considered as an adaline neural unit. Thus, a neurule base consists of a number of autonomous adaline units (neurules), expressed in a symbolic oriented syntax. There are two inference processes for neurules: the connectionism-oriented process, which gives pre-eminence to neurocomputing approach, and the symbolism-oriented process, which gives pre-eminence to a symbolic backwards chaining like approach. Symbolism-oriented process is proved to be more efficient than the connectionism-oriented one, in terms of the number of required computations (56.47% to 63.88% average reduction) and the mean runtime gain (59.95% to 64.89% in average), although both require almost the same average number of input values. The neurule-based explanation mechanism provides three types of explanations: 'how' a conclusion was derived, 'why' a value for a specific input variable was asked from the user and 'why-not' a variable has acquired a specific value. As shown by experiments, the neurule-based explanation mechanism is superior to that provided by known connectionist expert systems, another neuro-symbolic integration category. It provides less in number (64.38% to 69.28% average reduction) and more natural explanation rules, thus increasing efficiency (mean runtime gain of 56.65% to 56.73% in average) and comprehensibility of explanations.
Keywords: neuro-symbolic approaches; integrated intelligent systems; rule-based reasoning and explanation; hybrid intelligent systems; hybrid expert systems

1. Introduction

Most of the intelligent methods have advantages as well as disadvantages. Research in Artificial Intelligence (AI) has shown that approaches integrating (or combining) two or more intelligent methods may provide benefits (Tweedale & Jain, 2014; Hatzilygeroudis & Prentzas, 2011a). This is accomplished by exploiting the advantages of the integrated methods to overcome their disadvantages. Complementarity in advantages and disadvantages of the combined methods is usually the basis to the success of such integrations. Example types of popular integrations, among others, involve neuro-symbolic approaches, integrating neural networks with symbolic methods (Garcez d' Avila & Lamb, 2011; Hatzilygeroudis & Prentzas, 2004a), neuro-fuzzy approaches, integrating neural networks with fuzzy methods (Zhang, Ma, & Yang, 2015; Evans & Kennedy, 2014; Lin et al., 2012), approaches combining neural networks and genetic algorithms (Huang, Li, & Xiao, 2015) and approaches combining case-based reasoning with rule-based reasoning (Prentzas & Hatzilygeroudis, 2007) or other intelligent methods (Chuang & Huang, 2011; Prentzas & Hatzilygeroudis, 2009).

A number of neuro-symbolic formalisms have been introduced during last decade (Garcez d' Avila, Broda, & Gabbay, 2002; Garcez, D'Avila, Lamb, & Gabbay, 2009; Hatzilygeroudis & Prentzas, 2004a). Combinations of symbolic rules (of propositional type) and neural networks constitute a large proportion of neuro-symbolic approaches

(Gallant, 1993; Towell & Shavlik, 1994; Holldobler & Kalinke, 1994; Hatzilygeroudis & Prentzas, 2000, 2001). The efforts integrating rules and neural networks may yield effective formalisms by exploiting the complementary advantages and disadvantages of the integrated components (Hatzilygeroudis & Prentzas, 2004a). Symbolic rule-based systems possess positive aspects such as naturalness and modularity of the rule base, interactive reasoning process and ability to explain reasoning results. Neural networks lack the naturalness and modularity of symbolic rules and it is also difficult (or impossible) to provide explanations. Explanations are crucial in certain domains such as medicine and finance. However, symbolic rules have disadvantages such as, difficulty in acquiring rules from the experts (known as the 'knowledge acquisition bottleneck'), inability to draw conclusions when there are missing values in the input data and possible problems in cases of unexpected input values or combinations of them. On the other hand, neural networks provide generalization, representation of complex and imprecise knowledge and knowledge acquisition from training examples.

Neurules are a type of integrated rules combining symbolic rules and neurocomputing (Hatzilygeroudis & Prentzas, 2000, 2001). Neurules belong to neuro-symbolic representations resulting in a uniform, seamless combination of the two integrated components. Most of the existing such approaches give pre-eminence to connectionism. As a consequence, they do not offer important advantages of symbolic rules, like naturalness and modularity, and also do not provide interactive inference and explanation. Neurules follow a different direction by giving priority to the symbolic than the connectionist framework. Therefore, the knowledge base exhibits characteristics such as naturalness and modularity, to a large degree. Furthermore, neurule-based systems provide interactive inference and explanation.

Integration in neurules involves all knowledge representation aspects: syntax, semantics and reasoning. Hybridism in syntax and semantics has been presented in most of our past works on neurules and for the sake of completeness is briefly presented here too. Reasoning via neurules can be done via two different inference processes. The one gives pre-eminence to neurocomputing, namely *connectionism-oriented inference*, whereas the other to symbolic reasoning, namely *symbolism-oriented inference*. Both inference processes are integrated in their nature. Connectionism-oriented inference process has been presented in (Hatzilygeroudis & Prentzas, 2010) and compared to two alternative inference mechanisms used in connectionist expert systems (Gallant, 1993; Ghalwash, 1998), a type of neuro-symbolic systems. An initial version of the symbolism-oriented inference has been presented in (Hatzilygeroudis & Prentzas, 2000).

In this paper, we present an improved symbolism-oriented inference process. Improvement refers to the number of required computations to produce conclusions, the ability to work with any order of neurule conditions and the ability to work with two different sets of discrete values for representing 'true', 'false' and 'unknown' states. We present experimental results comparing the performance of the new symbolism-oriented process with the connectionism-oriented one.

However, the main contribution of this paper is the introduction of an explanation mechanism for neurule-based inference. The explanation mechanism provides three types of explanations: 'how', 'why' and 'why-not'. We also present experimental results comparing the 'how' explanation mechanism with the corresponding mechanism used in connectionist expert systems.

This paper is structured as follows. Section 2 briefly discusses related work. Section 3 presents neurules. Section 4 discusses the two alternative inference

processes. Section 5 presents the explanation mechanism. Section 6 presents explanation examples. Section 7 presents experimental results involving inference and explanation. Section 8 concludes.

2. Related work

The objective of our work is to remain on the symbolic ground and incorporate techniques from the connectionist approach into propositional type symbolic rules to improve their representation capabilities and performance, without significantly reducing features, like naturalness and modularity, or sacrificing functionalities, like interactive inference and explanation. Many attempts based on the connectionist ground, which simulate or translate symbolic processes within a neural network, have been made in the recent years. However, a few of them are able to provide any of the above features in a satisfactory degree.

We can specify two main research trends. The first trend stems from (Holldobler & Kalinke, 1994), where a way to translate a propositional logic program (i.e. a set of facts and rules) into a neural network, by encoding its associated semantic operators in a connectionist way, is introduced. However, it is not accompanied by any reasoning or explanation process. KBANNs (Knowledge-Based Artificial Neural Networks) (Towel & Shavlik, 1994) use a core of propositional rules to construct an initial neural network and then use empirical knowledge to train the network. This leads to more efficient training and refinement of the initial symbolic knowledge. Such approaches are also not accompanied by integrated reasoning or explanation processes. Outputs are produced from the trained network via neurocomputing (i.e. numeric computation) methods without any explanation.

The other trend concerns connectionist expert systems. In connectionist expert systems, domain concepts are associated with neural network nodes and relationships among concepts are associated with links among nodes. A representative such approach is MACIE (Gallant, 1993). MACIE is accompanied by an inference and an explanation mechanism. The performance of the MACIE inference engine was improved by the introduction of the recency inference engine (Ghalwash, 1998). A comparison of the time performance of these two inference engines, taking into account the required number of computations is presented in (Hatzilygeroudis & Prentzas, 2010). A disadvantage of connectionist expert systems is that their knowledge bases lack the naturalness and modularity of symbolic rule bases. Meaningless nodes are inserted by the training mechanism, to be able to handle non-linearity of data, thus making representation and provided explanations unnatural (see Hatzilygeroudis & Prentzas, 2001).

3. Neurules: Syntax and Semantics

Neurules are a kind of integrated rules. The form of a neurule is depicted in Fig.1a. Each condition C_i is assigned a number sf_i, called its *significance factor*. Moreover, each neurule itself is assigned a number sf_0, called its *bias factor*. Internally, each neurule is considered as an adaline unit neural (Fig.1b). The *inputs* C_i (i=1,...,n) of the unit are the conditions of the neurule. The weights of the unit are the significance factors of the neurule and its bias is the bias factor of the neurule. The existence of bias (bias factor) helps in training the adaline unit (neurule), more specifically helps the training algorithm in converging more easily and for more cases. Training a neural unit (neurule) means finding values for its weights (significance factors) that satisfy given data, i.e. classify them correctly. Each input takes a value from the following set of discrete values: [1 (true), -1 (false), 0 (unknown)].

The *output D*, which represents the conclusion (decision) of the neurule, is calculated via the standard formulas:

$$D = f(a),\ a = sf_0 + \sum_{i=1}^{n} sf_i C_i \quad (1)$$

$$f(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ -1 & \text{otherwise} \end{cases} \quad (2)$$

where ***a*** is the *activation value* and *f(x)* the *activation function*, which is a threshold function. Hence, the output can take one of two values ('-1', '1') representing failure and success of the rule respectively. The significance factor of a condition represents the significance (weight) of the condition in drawing the conclusion.

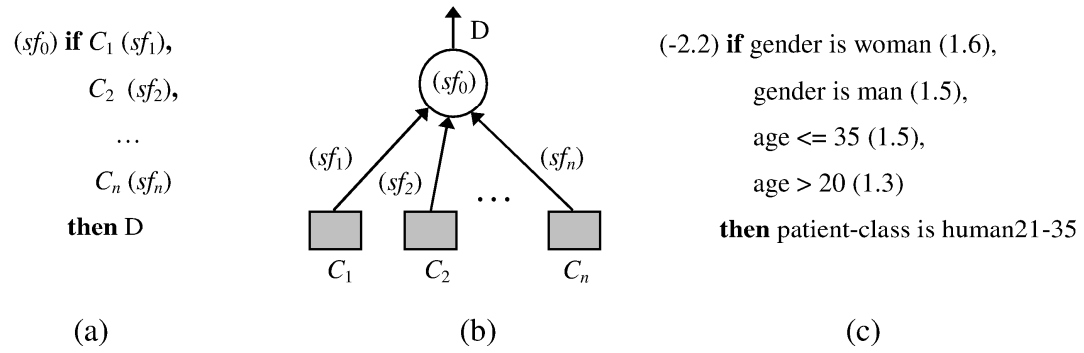

Fig. 1. (a) Form of a neurule, (b) a neurule as an adaline unit, (c) an example neurule

The general syntax of a neurule (in a BNF notation, where '< >' denotes non-terminal symbols) is:

<rule>::= (<bias-factor>) **if** <conditions> **then** <conclusion>
<conditions>::= <condition> | <condition>,<conditions>
<condition>::= <variable> <l-predicate> <value> (<significance-factor>)
<conclusion>::= <variable> <r-predicate> <value> .

where <variable> denotes a *variable*, that is a domain concept, e.g. 'gender', 'pain' etc in a medical domain, and <l-predicate> denotes a symbolic or a numeric predicate. The symbolic predicates are {is, isnot}, whereas the numeric predicates are {<, <=, >, >=, =, !=}. <r-predicate> can only be a symbolic predicate. <value> denotes a value; it can be a symbolic (e.g. "male", "night-pain") or a numeric value (e.g "5"). <bias-factor> and <significance-factor> are (real) numbers. For instance, in the condition 'fever isnot high (5.0)', 'fever' is a variable, 'isnot' is the symbolic predicate, 'high' is a symbolic value that variable 'fever' may take and '5.0' is the significance factor. Let us suppose that variable 'fever' can take a value from the set of values {low, medium, high}. Then this condition is satisfied in case 'fever' takes a value that is not 'high' (i.e. it is 'low' or 'medium'). An example simple neurule is presented in Fig. 1c.

Let us consider a finite set of *variables* V = $\{V_i\}$, $1 \leq i \leq m$, where each variable V_i can take values from a (corresponding) set of discrete values $S_{V_i} = \{v_{ij}\}$, $1 \leq j \leq k$. We distinguish three types of variables: *input or askable variables, intermediate or inferable variables* and *output or goal variables*. An input variable takes values from the user (input data). Intermediate or output variables take values through inference, since they represent intermediate and final conclusions respectively. The sets of input,

intermediate and goal variables are denoted by V_{INP}, V_{INT} and V_G respectively. We also distinguish between *input*, *intermediate* and *output neurules*. An input neurule is a neurule having only input variables in its conditions and intermediate or output variables in its conclusions. An intermediate neurule is a neurule having at least one intermediate variable in its conditions and intermediate variable in its conclusion. An output neurule is one having an output variable in its conclusion.

Neurules are produced either from symbolic rules (Hatzilygeroudis & Prentzas, 2000) or from empirical data in the form of training examples (Hatzilygeroudis & Prentzas, 2001). Each construction process creates at least one neurule for each intermediate or final conclusion. An adaline unit is initially created for each conclusion. A training set is also produced for each adaline unit. Each unit is individually trained via the Least Mean Square (LMS) algorithm. When the initial training set is inseparable, more than one neurule having the same conclusion are produced for the corresponding conclusion.

For example, the neurule in Fig. 1c has been produced from the following two symbolic rules:

if gender is woman
  and age <= 35
  and age > 20
then patient-class is human21-35

if gender is man
  and age <= 35
  and age > 20
then patient-class is human21-35

Also, it is easy to see that, if ‘gender is woman’, ‘age <= 35’ and ‘age > 20’ are true, hence ‘gender is man’ is false, then a = -2.2+1.6*1+1.5*(-1)+1.5*1+1.5*1 = 0.9, $f(a)$ = 1 (formulas 1 and 2), the neurule is fired and its conclusion is produced. This “simulates” firing of the left symbolic rule above. Similarly, if ‘gender is man’, ‘age <= 35’ and ‘age > 20’ are true, the neurule is again fired, “simulating” firing of the right symbolic rule above. So, a neurule acts as a composition of more than one symbolic rules that simulates their firing conditions. As we see later, use of a neuron to represent a rule gives us the opportunity to use special metrics to avoid evaluation of all of the conditions of a neurule to see if it fires or not, thus improving efficiency.

In the sequel, we use the following notation and definitions for a neurule R_k.

- n_{R_k} : the number of conditions of R_k.
- $sf_0^{R_k}$: the bias factor of R_k.
- $C_i^{R_k}$: the i^{th} condition of R_k, ($C_i^{R_k}$ ≡ “$V_i^{R_k}$ is $v_i^{R_k}$”)
- $V_i^{R_k}$: the variable involved in $C_i^{R_k}$
- $v_i^{R_k}$: the value involved in $C_i^{R_k}$.
- $sf_i^{R_k}$: the significance factor of $C_i^{R_k}$.
- D^{R_k} : the conclusion of R_k, (D^{R_k} ≡ “$V_d^{R_k}$ is $v_d^{R_k}$”)
- $V_d^{R_k}$: the variable involved in D^{R_k} .
- $v_d^{R_k}$: the value involved in D^{R_k} .
- $V_{C_k}^{R_k}$: the variable involved in condition C_k of R_k.
- *vars*(R_k): the set of variables involved in R_k, i.e.

$$vars(R_k) = \{V_1^{R_k}, V_2^{R_k}, \ldots, V_{n_{R_k}}^{R_k}, V_d^{R_k}\}.$$

- *conds*(R_k): the set of conditions of R_k, i.e.

$$conds(R_i) = \{ C_i^{R_k} , 1 \leq i \leq n_{R_k} \}.$$

The run-time part of a neurule-based system consists of four modules, functionally similar to those of a conventional rule-based system: *neurule base (NRB)*, *neurule-based inference engine*, *working memory (WM)* and *neurule-based explanation mechanism*. The neurule-based inference engine derives conclusions by using the neurules contained in NRB and *fact assertions* in WM. The neurule-based explanation mechanism provides explanations about made inferences.

A fact assertion is a pair (F_i, $ass(F_i)$) consisting of:

- a *fact* $F_i \equiv$ "V_{F_i} <l-predicate> v_{F_i} ", $V_{F_i} \in$ V, $v_{F_i} \in S_{V_{F_i}}$ and
- the assertion value $ass(F_i) \in$ {TRUE, FALSE, UNKNOWN} associated with fact F_i.

Facts have the same format as neurule conditions or conclusions. Fact assertions are either given by the user, if they involve input variables, or derived during inference, if they involve intermediate or output variables. WM also contains variable-value pairs of the form: "(V_{F_i}, v_{F_i})", where $V_{F_i} \in$ V, $v_{F_i} \in S_{V_{F_i}}$. WM = {(age, 28), ("gender is man", TRUE), ("pain is night-pain", TRUE)} may be a working memory instance.

Table 1. Example neurules

R_1 (-0.5) **if** gender is man (3.4), age <= 20 (3.3), gender is woman (3.0) **then** patient-class is human0-20	R_2 (-2.2) **if** gender is woman (1.6), gender is man (1.5), age <= 35 (1.5), age > 20 (1.3) **then** patient-class is human21-35
R_3 (-4.1) **if** age <= 55 (3.3), gender is woman (3.2), age > 35 (3.0), gender is man (2.6) **then** patient-class is human36-55	R_4 (-5.6) **if** patient-class is human21-35 (8.4), pain is night-pain (8.2), antinflam-reaction is medium (8.0), fever is no-fever (5.2), pain is continuous-pain (5.0), patient-class is human0-20 (4.7) **then** disease-type is arthritis
R_5 (-7.9) **if** patient-class is human36-55 (3.2), pain is continuous-pain (2.9), antinflam-reaction is high (2.8), joints-pain is no-pain (2.6) **then** disease-type is arthritis	R_6 (1.4) **if** fever is low (5.2), fever is high (5.2), patient-class is human36-55 (5.2), pain is night-pain (5.0), fever is medium (5.0), patient-class is human0-20 (4.5) **then** disease-type is primary-malignant
R_7 (-9.7) **if** antinflam-reaction is none (12.4), pain is night-pain (12.3), patient-class is human21-35 (11.9), pain is continuous-pain (8.8), patient-class is human0-20 (8.4), fever is no-fever (4.5) **then** disease-type is primary-malignant	

Table 1 presents some neurules related to the problem of diagnosis of bone diseases. They have been produced through conversion from corresponding symbolic rules (Hatzilygeroudis, Vassilakos, & Tsakalidis, 1997), which are simple rules used

for preliminary diagnosis, mostly based on demographic and clinical data (there are other rules that based on the preliminary decisions and special laboratory data make final diagnoses). Rules R1-R3 classify patients into one of three classes, depending on their gender and age (in the real system there are other classes too). They involve two input variables ('gender' and 'age') and an intermediate variable ('patient-class'), thus they are input rules. Rules R4-R7 make (preliminary) diagnoses mostly based on clinical data. They involve four input variables ('pain', 'fever', 'joints-pain' and 'antiflam-reaction'), an intermediate variable ('patient-class') and an output variable ('disease-type'), thus they are output rules. 'age' is a numeric variable, whereas the rest variables are categorical.

4. Integrated Inference Engine

The inference engine associated with neurules implements the way neurules co-operate to derive conclusions. It can support two alternative integrated inference processes. The one gives pre-eminence to neurocomputing, and is called *connectionism-oriented inference process*, whereas the other to symbolic reasoning, and is called *symbolism-oriented inference process*. In the connectionism-oriented process, the choice of the next rule to be considered is based on a neurocomputing measure, but the rest is symbolic. The connectionism-oriented approach has been presented in (Hatzilygeroudis & Prentzas, 2010). In the symbolism-oriented process, a type of a classical rule-based reasoning is employed, but evaluation of a rule is based on neurocomputing measures. The symbolism-oriented approach is an effort to rely on symbolic approaches, like the rule-based backward chaining approach. An initial version of the symbolism-oriented process has been presented in (Hatzilygeroudis & Prentzas, 2000). In that version of the symbolism-oriented process, organization of neurule conditions as well as inference heuristics were accustomed to the set of discrete input values [1 (true), 0 (false), 0.5 (unknown)].

In this section, we present an improved version of the symbolism-oriented approach and experimentally compare it with the connectionism-oriented one. We also discuss issues involving use of alternative sets of input values, [1 (true), -1 (false), 0 (unknown)] and [1 (true), 0 (false), 0.5 (unknown)]. The neurule base construction mechanisms have been modified to be able to work with values of either set in the training process. For instance, an equivalent neurule base to the one presented in Table 1 can be easily constructed using the {1, 0, 0.5} set of values. Obviously, the set of values used to construct a neurule base is also used for drawing conclusions during inference.

4.1 Evaluation Aspects

Neurules are evaluated during reasoning based on evaluation of their conditions. Aspects involving condition and neurule evaluation are discussed in this section.

Conditions are evaluated during reasoning based on fact assertions and variable-value pairs contained in WM. A condition C_i can evaluate to TRUE/FALSE/UNKNOWN (denoted by $ass(C_i)$ = TRUE/FALSE/UNKNOWN). Evaluation of a condition $C_i \equiv$ "V_i is v_i" may enable evaluation of its 'sibling' conditions that is, conditions of the form $C_j \equiv$ "V_i is v_j", where $v_i, v_j \in S_{V_i}$ and $v_i \neq v_j$. More specifically, if $ass(C_i)$ = TRUE, then $ass(C_j)$ = FALSE, since a variable cannot take more than one value simultaneously. For example, if "fever is low" is TRUE, then "fever is high" is FALSE. Furthermore, if $ass(C_i)$ = UNKNOWN, then $ass(C_j)$ =

UNKNOWN, since 'unknown' means that there is no evidence about any value of a variable. A similar process may be followed for other types of condition predicates.

Each evaluated condition $C_i^{R_k}$ is assigned a numeric value, denoted by $assv(C_i^{R_k})$, corresponding to TRUE, FALSE or UNKNOWN. The value $assv(C_i^{R_k})$ depends on the used set of input values mentioned above. If the set of [1 (true), -1 (false), 0 (unknown)] is used, the value $assv(C_i^{R_k})$ is computed according to Eq. (3).

$$assv(C_i^{R_k}) = \begin{cases} 1 & \text{if} \quad ass(C_i^{R_k}) = \text{TRUE} \\ -1 & \text{if} \quad ass(C_i^{R_k}) = \text{FALSE} \\ 0 & \text{if} \quad ass(C_i^{R_k}) = \text{UNKNOWN} \end{cases} \tag{3}$$

If the set of [1 (true), 0 (false), 0.5 (unknown)] is used, the value $assv(C_i^{R_k})$ is computed according to Eq. (4).

$$assv(C_i^{R_k}) = \begin{cases} 1 & \text{if} \quad ass(C_i^{R_k}) = \text{TRUE} \\ 0 & \text{if} \quad ass(C_i^{R_k}) = \text{FALSE} \\ 0.5 & \text{if} \quad ass(C_i^{R_k}) = \text{UNKNOWN} \end{cases} \tag{4}$$

Using {1, 0, 0.5} as the set of input values, instead of {1, -1, 0}, gives a more natural representation of the contribution of each condition to the conclusion of a neurule as will be clear in the following.

Conditions are considered for evaluation, when required by the reasoning process. The conditions of each neurule are distinguished into *evaluated* and *unevaluated* conditions. The set of evaluated conditions of a neurule R_k is denoted by $\mathrm{C}_{\mathrm{E}}^{R_k}$ and that of unevaluated by $\mathrm{C}_{\mathrm{U}}^{R_k}$.

According to formulas 1 and 2 (Section 3), evaluation of a neurule R_k requires evaluation of all its conditions. However, based on the values of two metrics of R_k, the *known sum*, denoted by $ks(R_k)$, and the *remaining sum*, denoted by $rs(R_k)$, a neurule may be evaluated without having evaluated all of its conditions, fact that speeds up inference.

$ks(R_k)$ represents the contribution of the (already) evaluated conditions of R_k in drawing its conclusion (i.e. in activating the corresponding adaline unit), which is expressed as the sum of the contributions of each condition. The contribution of each evaluated condition is the product of its significance factor $sf_i^{R_k}$ and its value $assv(C_i^{R_k})$ (see Formula 5).

$rs(R_k)$ represents the largest possible contribution of the unevaluated conditions of R_k in drawing its conclusion (i.e. in activating the corresponding adaline unit), if they were evaluated. It is expressed as the sum of the largest possible contribution of each unevaluated condition, which is the absolute value of its significance factor $|sf_i^{R_k}|$, given that its maximum absolute value is 1 (see Formula 6).

$$ks(R_k) = sf_0^{R_k} + \sum_{C_i^{R_k} \in \mathrm{C}_{\mathrm{E}}^{R_k}} sf_i^{R_k} \; assv(C_i^{R_k}) \tag{5}$$

$$rs(R_k) = \sum_{C_i^{R_k} \in \mathrm{C}_{\mathrm{U}}^{R_k}} \left| sf_i^{R_k} \right| \tag{6}$$

An evaluated neurule is either *fired* or *blocked*. In the first case, the neurule has evaluated to '1' and in the second case to '–1'. The set of fired rules during an inference process is denoted by R_F, whereas that of blocked rules by R_B. Also, the set of evaluated rules is denoted by R_E and that of unevaluated by R_U. Furthermore, the set of variables that have and have not acquired values during reasoning are denoted by V_E and V_U, respectively.

Based on $ks(R_k)$ and $rs(R_k)$, one may define the firing and blocking conditions of a neurule R_k (Hatzilygeroudis & Prentzas, 2010). More specifically, the *firing condition* of R_k is the situation where $|ks(R_k)| \geq rs(R_k)$ and $ks(R_k) \geq 0$. When it is satisfied, R_k is fired and its conclusion is produced. The *blocking condition* of R_k is the situation where $|ks(R_k)| > rs(R_k)$ and $ks(R_k) < 0$. When it is satisfied, R_k is blocked. For proofs refer to (Hatzilygeroudis & Prentzas, 2010).

The firing and blocking conditions hold for both sets of input values ({1, 0, 0.5} and {1, -1, 0}). However, more "fine-grained" remaining sums can be defined, especially for the set {1, 0, 0.5}, which may speed up evaluation of neurules. For this purpose, the following are taken into account:

- Conditions evaluated to FALSE do not contribute in the activation value of a neurule.
- Conditions evaluated to TRUE/UNKNOWN and having positive significance factors positively (negatively) contribute in firing (blocking) of a neurule.
- Conditions evaluated to TRUE/UNKNOWN and having negative significance factors negatively (positively) contribute in firing (blocking) of a neurule.

Therefore, we define more "fine-grained" remaining sums as follows:

Definition 1. The *positive remaining sum* of a neurule R_k, denoted by $posrs(R_k)$, is defined as the largest possible positive weighted sum of the unevaluated conditions of it contributing in its firing:

$$posrs(R_k) = \sum_{C_i^{R_k} \in C_U^{R_k},\, sf_i^{R_k} > 0} sf_i^{R_k} \tag{7}$$

Definition 2. The *negative remaining sum* of a neurule R_k, denoted by $negrs(R_k)$, is defined as the largest possible negative weighted sum of the unevaluated conditions of it contributing in its blocking:

$$negrs(R_k) = \sum_{C_i^{R_k} \in C_U^{R_k},\, sf_i^{R_k} < 0} sf_i^{R_k} \tag{8}$$

Obviously $rs(R_k) = posrs(R_k) + |negrs(R_k)|$. We may also give the following more "fine-grained" definitions for success (or firing) condition and failure (or blocking) condition of a neurule that hold only for the set of {1, 0, 0.5}:

Definition 3. *Firing condition* of a neurule R_k is the situation where $|ks(R_k)| \geq |negrs(R_k)|$ and $ks(R_k) \geq 0$.

Definition 4. *Blocking condition* of a neurule R_k is the situation where $|ks(R_k)| > posrs(R_k)$ and $ks(R_k) < 0$.

The rationale behind the above two definitions is that always $posrs(R_k) \geq 0$ and $negrs(R_k) \leq 0$. So, the contribution of the corresponding conditions in the activation value of the neurule is known beforehand to be positive (or zero) and negative (or zero) respectively.

Neurule conditions are ordered in a way that results in more efficient computations during inference. Experiments have shown that this is achieved when neurule conditions are arranged according to the descending order of the absolute values of their significance factors.

Both inference procedures use a goal stack containing *goal facts* G_i that is, the neurule conclusions including a goal variable. A goal fact G_i is defined as follows:

$$G_i \equiv \text{“}V_{G_i} \text{ is } v_{G_i}\text{”, where } V_{G_i} \in \mathrm{V_G},\ v_{G_i} \in S_{V_{G_i}}$$

4.2 Symbolism-oriented process

The symbolism-oriented inference process is based on a backward chaining strategy. There are two stacks used, a *goal stack* (*GS*) containing the goal facts, where the *current goal* (G_c) is always on its top, and a *rule stack* (*RS*) containing the rules involved in the current inference session, where the *current rule* (R_c) under evaluation is always on its top.

The conflict resolution strategy, due to backward chaining and the neurules, is based on textual order. Inference stops successfully, when one or more output rules have fired and the goal stack is empty. It stops unsuccessfully, when there are no facts (conclusions) in WM containing goal variables assigned the TRUE value and no further action can be taken.

The symbolism-oriented process has been improved compared to the initial version presented in (Hatzilygeroudis & Prentzas, 2000) according to the following aspects:

- In the initial version, the conditions of neurules were organized into two groups, the group of conditions with negative significance factors and the group of conditions with positive significance factors. This ordering was accustomed to the set of discrete values {1, 0, 0.5}, but the current ordering (mentioned in the previous section) may lead to more efficient inferences.
- In the initial version, inference heuristics were accustomed to the set of input values {1, 0, 0.5} and to the specific ordering of conditions. Current version may work for any ordering of conditions, although inference is more efficient for the ordering of conditions mentioned in Section 4.1. Furthermore, the symbolism-oriented process presented in this paper may work for both sets of input values ({1, -1, 0} and {1, 0, 0.5}) using the remaining sum to define the firing or blocking condition of a neurule. For the set {1, 0, 0.5}, inference is more efficient if the positive and negative remaining sums are used. In either case, inference is more efficient for the set {1, 0, 0.5} compared to (Hatzilygeroudis & Prentzas, 2000). The inference process presented in (Hatzilygeroudis & Prentzas, 2000) is identical to the process using positive and negative remaining sums in case the neurule conditions in the neurule base contain only positive significance factors.

A high-level description of the basic symbolism-oriented algorithm is as follows:

1. *Set the goal(s) on GS, the initial facts in the WM and compute (initial) ks, posrs and negrs for each neurule.*
2. *For each goal on GS do*

2.1. Find all neurules whose conclusions match the goal and put them on RS
2.2. For each neurule on RS do
2.2.1. Evaluate each condition of the neurule:
2.2.1.1. If the variable of the condition has got a value from evaluation of a previous neurule, just update ks, posrs and negrs of the rule.
2.2.1.2. If the condition contains an input variable, ask the user for a value and update ks, posrs and negrs of the rule.
2.2.1.3. If the condition contains an intermediate variable, make it the current goal and go to step 2.1.
2.2.2. After evaluation of each condition
2.2.2.1. If the firing condition is satisfied, update WM with the sibling assertions of the conclusion of the rule.
2.2.2.2. If the blocking condition is satisfied and the rest sibling rules are blocked, update WM with the fact corresponding to the falsity of the conclusion of the rule.
3. If WM contains any goal facts assigned a true value, return those goal facts; otherwise return failure.

4.3 Connectionism-oriented process

For the sake of completeness, we also present here a short description of the connectionism-oriented inference process.

The connectionism-oriented inference process focuses on the firing potential of each neurule, which is measured by a specific metric, called *firing ratio* (*fr*), related to the above mentioned known and remaining sums (see Hatzilygeroudis & Prentzas, 2010). Initially, *fr*s of all neurules are computed. In this process, after evaluation of a condition of a neurule, the *fr*s of the neurules that contain that variable (called *affected rules*) are updated. In the next step, the neurule with the maximum *fr* is considered, given that it is the neurule most likely (i.e. with the greatest intention) to fire. This means that its first unevaluated condition is considered as the next goal and so on. As soon as a neurule is fired, the WM is updated with the corresponding conclusion. A similar thing happens when a neurule is blocked. So, at each step of the process, rules are competing between each other towards which is closer to be evaluated, based on their *fr*s. The one with the greatest *fr* takes the lead. Inference stops either successfully, when there are facts in WM containing goal variables and assigned the TRUE value and no further action can be taken, or unsuccessfully, otherwise.

5. Explanation Mechanism

The explanation mechanism is used to provide some type of explanation related to an inference session. More specifically, the explanation mechanism associated to neurules provides the following types of explanation:

- *How*: explanation of how a conclusion has been derived.
- *Why-not*: explanation of why an inferable (intermediate or output) variable has not acquired a specific value.
- *Why*: explanation of why is required to give value for a particular input variable.

In this section, the processes concerning the above explanation types are presented.

5.1 'How' Explanation Process

The 'how' explanation mechanism justifies inferences, by producing a set of simple if-then rules explaining how a conclusion has been drawn. To this end, an initial explanation rule for a final conclusion is created based on some (the necessary) of the

conditions of the corresponding fired neurule that produced that conclusion. Afterwards, additional explanation rules are created for the necessary conditions of the initial explanation rule that contain intermediate variables, and so on. The process stops when the last explanation rules have conditions including only input variables. So, finally, we end up with a set of if-then rules whose conclusions correspond to the final conclusion(s) drawn and to the intermediate conclusions contributing in drawing the final conclusion(s). The conditions of the explanation rules correspond to a subset of the conditions that participated in drawing the conclusion(s). Here is a structured description of the corresponding algorithm:

1. *For each neurule that has been fired and has a goal variable in its conclusion do*
   - 1.1. *Produce the corresponding explanation rule based on the conditions that were necessary for its evaluation.*
   - 1.2. *For each condition of the produced explanation rule that includes an intermediate variable do*
     - 1.2.1. *Find the neurule that produced the condition and set it as the current rule.*
     - 1.2.2. *Produce the explanation rule corresponding to the current rule as in step 1.1 and go to step 1.2*
2. *Transform conditions and conclusions of explanation rules by adapting their predicates.*
3. *Reorder produced rules and change their presentation for naturalness reasons.*

And here is a more formal description of the algorithm:

1. $HR \leftarrow \varnothing$
2. For each $R_k \in (R_G \cap R_F)$ do
2.1 $HR_k \leftarrow$ Produce_explanation_rule(R_k)
2.2 $HR \leftarrow HR \cup \{HR_k\}$
2.3 Produce_intermediate_explanation_rules(HR_k)
3. For each $HR_k \in HR$ do
3.1 conds(HR_k) $\leftarrow$ Transform_conds(HR_k)
3.2 Transform_concl(HR_k)
4. Reorder produced rules and change their presentation for naturalness reasons.

where *HR* is the set of produced explanation rules and HR_k the explanation rule produced from neurule R_k.

To be able to produce explanation rules, we introduce some new concepts. First, the conditions of an evaluated neurule are distinguished in *positive* and *negative conditions*. This distinction is based on the positive or negative contribution of the conditions in evaluating the corresponding neurule. There are different definitions for the positive and negative conditions according to whether the corresponding evaluated neurules were fired or blocked and also according to whether {1, -1, 0} or {1, 0, 0.5} set of input values is used. The following definitions are associated with the {1, -1, 0} input values set.

Definition 5. The *positive conditions* of an evaluated neurule R_k, denoted by *posConds*(R_k), are those that positively contributed in evaluation of the neurule. A condition $C_i^{R_k}$ positively contributes, if the product of its value with its significance

factor ($assv(C_i^{R_k}) * sf_i^{R_k}$) is positive, in case of a fired neurule, or negative, in case of a blocked neurule.

Definition 6. The *negative conditions* of an evaluated neurule R_k, denoted by *negConds*(R_k), are those that negatively contributed in evaluation of the neurule. A condition $C_i^{R_k}$ negatively contributes, if the product of its value with its significance factor ($assv(C_i^{R_k}) * sf_i^{R_k}$) is negative, in case of a fired neurule, or positive, in case of a blocked neurule.

Conditions that are unevaluated or negative are not included in the explanation rules, because they did not contribute or it is not clear how they contributed in drawing conclusions. Furthermore, some of the positive conditions may be also not included, based on the fact that they were not necessary in evaluating some neurule. More specifically, the conditions of an explanation rule are the ones having the most positive contribution in evaluating the corresponding neurule, with possible changes to their predicates. We call such conditions the *necessary conditions* of an evaluated neurule R_k. Respectively, *unnecessary conditions* of an evaluated neurule R_k are the positive conditions with the least positive contribution in the neurule's evaluation. The sets of *necessary* and *unnecessary conditions* of an evaluated neurule R_k are denoted by *necConds*(R_k) and *unnecConds*(R_k) respectively.

So, production of an explanation rule consists in determining the necessary conditions. This is based on the following approach, which tries to trace back the inference chain in an indirect way and reveal (or in another sense reconstruct) the equivalent symbolic rules that were fired. We introduce two sums, the *positive sum* of a neurule R_k, denoted by *pos-sum*(R_k), and its *negative sum*, denoted by *neg-sum*(R_k).

The positive sum is initially defined as the aggregate contribution value of all positive conditions:

$$pos\text{-}sum(R_k) = sf_0^{R_k} + \sum_{posConds} sf_i^{R_k} assv\left(C_i^{R_k}\right)$$

whereas the negative sum as the sum of the absolute values of the significance factors of both the negative and unevaluated conditions:

$$neg\text{-}sum(R_k) = \sum_{negConds} \left|sf_i^{R_k}\right| + \sum_{V_i^{R_k} \in V_U} \left|sf_i^{R_k}\right|$$

At this stage, |pos-sum(R_k)| > |neg-sum(R_k)|, due to firing condition. Also, we set *necConds*(R_k) = *posConds*(R_k) and *unnecConds*(R_k) = ∅. A repetitive process is then carried out. Successively, the condition with the least absolute significance factor in the current *necConds*(R_k) (denoted by $C_m^{R_k}$) is found and its contribution is subtracted from the *pos-sum*(R_k) and added to *neg-sum*(R_k). This is carried out as long as the absolute value of *pos-sum*(R_k) remains greater than the absolute value of *neg-sum*(R_k). Each time *necConds*(R_k) is updated by removing $C_m^{R_k}$. The remaining set of necessary conditions contains the conditions to be included in the corresponding explanation rule. The above is what '*Produce_explanation_rule(R_k)*' does.

Finally, for each condition of an explanation rule containing an intermediate variable an explanation rule is produced, in the same way as above. This is what '*Produce_intermediate_explanation_rules(HR_k)*' does.

Explanation rules produced as above described need a further refinement. Initially, significance factors of the conditions need not to be shown, as they will not play any role in the process, and should be removed. Also, conditions not providing useful information should be eliminated and some predicates may change. We present here such cases.

As a working example, suppose that $V_i^{R_k}$ ≡ "fever" having S_{V_i} = {low, medium, high} and conditions $C_1^{R_k}$ ≡ "fever is low", $C_2^{R_k}$ ≡ "fever is medium" and $C_3^{R_k}$ ≡ "fever is high" are present in *necConds*(R_k). Also, let us suppose that $ass\left(C_1^{R_k}\right) = TRUE$, $ass\left(C_2^{R_k}\right) = FALSE$, $ass\left(C_3^{R_k}\right) = FALSE$. In this case, the predicates of conditions $C_2^{R_k}$ and $C_3^{R_k}$ after transformation would become 'isnot', given their falsity. So, we will have the following subset of conditions {"fever is low", "fever isnot medium" and "fever isnot high"} in *necConds*(R_k). From these, only "fever is low" provides useful information in the explanation rule. So, this subset of sibling conditions is replaced by a single condition ("fever is low"). A similar process is followed in case of conditions with numeric predicates.

Furthermore, another case is when we have in *necConds*(R_k) a set of sibling conditions that cover all values of a specific variable but one and they are all false. Suppose that the above conditions $C_2^{R_k}$ and $C_3^{R_k}$ are in *necConds*(R_k), whereas $C_1^{R_k}$ is not. Conditions $C_2^{R_k}$ and $C_3^{R_k}$ cover two of the three values in S_{V_i} and, given their falsity, their predicate will become 'isnot', forming the following subset: {"fever isnot medium" and "fever isnot high"}. It is obvious that $C_2^{R_k}$ and $C_3^{R_k}$ can be removed and replaced by condition $C_1^{R_k}$. The above is what *Transform_conds(*HR_k*)* and *Transform_concl(*HR_k*)* do.

In case that the {1, 0, 0.5} set of input values is used, slightly different definitions are given for positive and negative conditions and also a slightly different process for explanation rule production is followed.

5.2 'Why-not' Explanation Process

The 'why-not' explanation type explains why a particular intermediate or output variable *V* has not acquired a specific value *v*. The explanation is again in the form of if-then rules. The generation of the explanation rules resembles the way the explanation rules of 'how' type are generated. More specifically, the explanation mechanism focuses on the blocked neurules having a conclusion containing variable *V* and the specific value *v*. If there are no blocked neurules having a conclusion containing the variable and the value, it means that reasoning was carried out without having to evaluate such neurules. However, such neurules will need to be evaluated in retrospect in order to produce 'why-not' explanations.

Following is a structured description of the corresponding algorithm. Actually, the main body is almost the same as that of 'how' explanation.

1. *Examine the blocked neurules whose conclusion contains* variable *V* and value *v.*
   1.1. If no such blocked neurule exists, evaluate each such (unevaluated) neurule.
2. *For each such blocked neurule, execute steps 1.1, 1.2, 2 and 3 of 'how' explanation process (structured description).*

Following is a more formal outline of the 'why-not' explanation mechanism, which tries to explain why variable *V* has not acquired value *v*.

1. HR $\leftarrow \varnothing$

2. If there is no $R_k \in R_B$ *with* $V_d^{R_k} \equiv V$ *and* $v_d^{R_k} \equiv v$ *do*

2.1 Evaluate all R_k *with* $V_d^{R_k} \equiv V$ *and* $v_d^{R_k} \equiv v$

3. For each $R_k \in R_B$ *with* $V_d^{R_k} \equiv V$ *and* $v_d^{R_k} \equiv v$ *do*

3.1 $HR_k \leftarrow$ *Produce_explanation_rule(*R_k*)*

3.2 $HR \leftarrow HR \cup \{HR_k\}$

3.3 Produce_intermediate_explanation_rules(HR_k*)*

4. Reorder produced rules and adjust their presentation for naturalness reasons.

5.3 'Why' Explanation Process

The 'why' explanation type explains why the system asks the user to provide a value for an input variable. The need for providing a value for an input variable arises when the inference mechanism is unable to evaluate a neurule condition containing that variable based on the data in WM. That condition is the unevaluated condition of the corresponding neurule with the maximum absolute significance factor. The 'why' explanation mechanism allows the user to ask "why?" and gives as an answer a tracing of the corresponding steps of the inference process. In other words, it returns the sequence of the names of conclusion variables of the unevaluated neurules that caused asking for a value of the input variable alongside some additional information.

In a simple case, only one rule is involved. This happens when the input variable under consideration was included in the first condition to be evaluated. In a more complex case, more than one rule is involved. In such cases, one or more conditions with intermediate variables are involved, so that a chain of rules is created, where the last one includes the condition with the input variable under consideration.

Following is a high-level presentation of the output produced by the 'why' explanation process for the set of involved neurules $\{R_1, R_2, ..., R_{n-2}, R_{n-1}, R_n\}$. In this case, the inference process has stopped at neurule R_n and the conclusion of each neurule R_i $(1 \leq i \leq n\text{-}1)$ is the first unevaluated condition of each neurule R_{i+1}, whereas the first unevaluated condition of neurule R_1 contains the corresponding variable and value.

"Decision on $V_d^{R_n}$ *requires decision on* $V_d^{R_{n-1}}$*.*

...

Decision on $V_d^{R_2}$ *requires decision on* $V_d^{R_1}$*.*

Decision on $V_d^{R_1}$ *requires value of* $V_i^{R_1}$*."*

The algorithm for the 'why' explanation mechanism requires a doubly linked list (denoted by *WHY-LIST*) containing the rules involved. More specifically, in case that the unevaluated condition with the maximum absolute significance factor of the current rule cannot be evaluated based on the data in the WM, the rule is inserted at the head of *WHY-LIST*. In the symbolism oriented process, *WHY-LIST* is handled a bit differently compared to the connectionism oriented process. In the symbolism oriented process, only the rule that is evaluated is removed from *WHY-LIST*. In the connectionism oriented process, the selected rule with the maximum *fr* determines which rules will be removed from *WHY-LIST*. In case the selected rule was not in *WHY-LIST*, all rules are removed from *WHY-LIST*. In case that the selected rule was in *WHY-LIST*, all rules inserted after it in *WHY-LIST* are removed from *WHY-LIST*.

To facilitate implementation, there are pointers to the head as well as the tail of the list.

Following is a more structured description of the corresponding algorithm:

1. *For each neurule that is on WHY-LIST starting from its tail do*
   1.1. *If the neurule is not on the head of the WHY-LIST, produce the corresponding explanation showing the dependence of the neurule's conclusion variable on the conclusion variable of the previous neurule (towards the head).*
   1.2. *Else (i.e. the neurule is on the head of the WHY-LIST), produce corresponding explanation showing the dependence of the neurule's conclusion variable on its condition's variable whose value is asked for.*

6. Explanation Examples

In the following, we provide certain examples concerning the aforementioned explanation processes. The explanation examples are based on results produced by the symbolism-oriented inference for the (partial) neurule base shown in Table 1.

Let us suppose that the following values are given for the input variables (in the order specified): (pain, night-pain), (fever, no-fever), (antinflam-reaction, none), (gender, man) and (age, 30) and stored in WM. Given these inputs, neurules R_4, R_5 and R_6 are blocked, whereas neurules R_2 and R_7 are fired. So, the following conclusions are derived: {"patient-class is human21-35", "disease-type is primary-malignant"}.

6.1 'How' explanation example

In this section, we present the 'how' type explanation that will be produced for the aforementioned inference example. We give tracings of them.

The explanation mechanism should provide explanations for the output conclusion "disease-type is primary-malignant" and the intermediate conclusion "patient-class is human21-35". In order to generate an explanation rule for the produced output conclusion, the explanation mechanism will examine neurule R_7, which has the corresponding output and was evaluated to '1' (TRUE). The explanation rule is produced as follows.

The set of positive conditions for R_7 are $posConds(R_7)$ = {'antinflam-reaction is none', 'pain is night-pain', 'patient-class is human21-35'}. The other three conditions of the neurule did not need to be evaluated, as resulted during inference. The tracing of the generation of the above explanation rule is as follows:

$necConds(R_7) \leftarrow$ {'antinflam-reaction is none', 'pain is night-pain', 'patient-class is human21-35'}
$unnecConds(R_7) \leftarrow \varnothing$
$pos\text{-}sum(R_7)$: -9.7 + 12.4 + 12.3 + 11.9 = 26.9
$neg\text{-}sum(R_7)$: 8.8 + 8.4 + 4.5 = 21.7

Since $|pos\text{-}sum(R_7)| > |neg\text{-}sum(R_7)|$, a check is made whether condition 'patient-class is human21-35', which is the one with the least absolute significance factor, can be removed. However, in that case $|pos\text{-}sum(R_7)|$ would become less than $|neg\text{-}sum(R_7)|$. So, finally $necConds(R_7)$ contains all three above conditions. Thus, the following explanation rule is produced:

EXR1
if antinflam-reaction is none
and pain is night-pain

and patient-class is human21-35
then disease-type is primary-malignant

Due to the fact that the explanation rule EXR1 contains an intermediate variable (i.e. 'patient-class'), another explanation rule should be generated for it (having as conclusion 'patient-class is human21-35'). To this end, the fired neurule R_2 is examined. The sets of positive and negative conditions for R_2 are $posConds(R_2)$ = {'gender is man', 'age <= 35', 'age > 20'} and $negConds(R_2)$ = {'gender is woman'}, respectively. The tracing of the generation of the above explanation rule is as follows:

$necConds(R_2) \leftarrow$ {'gender is man', 'age <= 35', 'age > 20'}
$unnecConds(R_7) \leftarrow \varnothing$
$pos\text{-}sum(R_2)$: -2.2 + 1.5 + 1.5 + 1.3 = 2.1
$neg\text{-}sum(R_2)$: 1.6

Since $|pos\text{-}sum(R_2)| > |neg\text{-}sum(R_2)|$, a check is made whether condition 'age > 20', which is the one with the least absolute significance factor, can be removed. However, in that case $|pos\text{-}sum(R_2)|$ would become less than $|neg\text{-}sum(R_2)|$. So, finally $necConds(R_2)$ contains all the three mentioned conditions and the following explanation rule is generated:

EXR2
if gender is man
and age <= 35
and age > 20
then patient-class is human21-35

To improve the naturalness of derived explanation rules, they are presented as follows (step 4 of the formal description of the algorithm):

disease-type is primary-malignant
because antinflam-reaction is none
and pain is night-pain
and patient-class is human21-35

patient-class is human21-35
because gender is man
and age < 36
and age > 20

6.2 'Why-not' explanation example

In this section, we present a 'why-not' explanation example produced by our system for the above inference. The system will explain why variable 'patient-class' has not acquired the value 'human0-20'. The explanation mechanism will focus on neurule R_1 whose conclusion is 'patient-class is human0-20'. This neurule was not evaluated during inference and therefore the 'why-not' explanation process will first have to evaluate it. Given the corresponding inputs, the neurule is blocked. Afterwards, explanation rule EXR3 is generated.

EXR3
if gender is man

and age > 20
then patient-class isnot human0-20

which is presented as

patient-class isnot human0-20
because gender is man
and age > 20

6.3 'Why' explanation example

In this section, we present a 'why' explanation produced by our system for the above inference. The system will explain why the user was asked to provide a value for variable 'gender'. In this case, the inference process stopped at neurule R4, whose first unevaluated condition (i.e. 'patient-class is human21-35') contained an intermediate variable. Neurule R2 contains a conclusion with the same intermediate variable. The unevaluated condition of R2 with the maximum significance factor contained the variable 'gender'. R2 causes the system to ask the user the value of variable 'gender'. Therefore, the system will generate the following explanation:

*"Decision on **disease-type** requires decision on **patient-class**.*
*Decision on **patient-class** requires **value of gender**."*

7. Experimental Results and Discussion

In this section, experimental results regarding the performance of neurules are presented with the help of a number of conducted experiments. These refer to the following:

- The symbolism-oriented process is compared to the connectionism-oriented process for the alternative sets of input values, {1, -1, 0} and {1, 0, 0.5}.
- For the set {1, 0, 0.5}, alternative versions of the symbolism-oriented inference process are compared.
- The neurule-based explanation mechanism is compared to the explanation mechanism used in connectionist expert systems, as presented in (Gallant, 1993) and (Ghalwash, 1998). Comparison is performed for both sets of input values.

To this end, we used four datasets and two symbolic rule bases. All involved mechanisms were implemented in C using the MS Visual Studio Express Edition. The experiments were run on a computer having a CoreI2 Duo CPU, 4 GB RAM and MS Windows 7 installed.

The four datasets satisfy at least one of the following requirements: (a) the involved attributes are categorical and (b) there is dependency information available. The first dataset (called ACUTE) is taken from (Gallant, 1993). The dataset is incomplete. It consists of 8 input data patterns out of 64 possible. For most of the input combinations no final conclusion can be drawn since all the output cells of the connectionist knowledge base become false and all the output rules of the corresponding neurule base become blocked. We give results for 27 input combinations for which final conclusion(s) can be drawn that is, one or more output cells of the connectionist knowledge base become true and one or more output rules of the corresponding neurule base fire. The second dataset (called LENSES) is taken from the UCI Machine Learning Repository (Bache & Lichman, 2013). This dataset is complete and contains 24 input patterns each consisting of four input and one output

variable. The third and fourth datasets (called CAR and NURSERY respectively) are also from the UCI Machine Learning Repository. However, we also used the dependency information provided by the donators of the datasets (Bohanec & Zupan, 1997), which is not included in the Repository. The CAR dataset with the available dependency information includes six input, three intermediate and one output variable. The NURSERY dataset with the available dependency information includes eight input, four intermediate and one output variable. The four datasets were used to construct corresponding four neurule bases according to the mechanism presented in (Hatzilygeroudis & Prentzas, 2001). Furthermore, four equivalent knowledge bases used in connectionist expert systems were constructed according to the instructions outlined in (Gallant, 1993).

The two symbolic rule bases we used involve two different domains. One rule base, mentioned hereafter as CSRB, concerns a banking domain (i.e. credit scoring), specifically assessment of bank loan applicants (Hatzilygeroudis & Prentzas, 2011b). In (Hatzilygeroudis & Prentzas, 2011b), we present in detail the design, implementation and evaluation of two separate intelligent systems that assess bank loan applications, a fuzzy and a neuro-symbolic expert system. The former employs fuzzy rules based on knowledge elicited from experts. The domain concerns five input variables, two intermediate variables and an output variable. Each input and intermediate variable takes three discrete values whereas the output variable takes five discrete values. We used one-hundred-seventy-three (173) of the constructed rules in our experiments and in the conversion process we assumed that values are non-fuzzy.

The other rule base, mentioned hereafter as MDRB, concerns a medical domain and more specifically bone disease diagnosis (Hatzilygeroudis, Vassilakos, & Tsakalidis, 1997). It contains one-hundred-thirty-four (134) symbolic rules. The rules involve five input variables and one output variable. The output variable takes fifty-nine (59) discrete values. Four of the input variables take four discrete values and one input variable takes five discrete values.

The two rule bases were converted to two equivalent neurule bases using the conversion mechanism presented in (Hatzilygeroudis & Prentzas, 2000), which merges symbolic rules having the same conclusion into neurules. Two corresponding connectionist knowledge bases were also constructed according to the guidelines described in (Gallant, 1993). For this purpose, we used: (a) the dependency information inherent in each group of all symbolic rules having the same conclusion and (b) the training sets extracted from the truth table of the combined logical functions of each rule group.

For each dataset and rule base, two neurule bases and two connectionist knowledge bases were constructed, one corresponding to set {1, -1, 0} and one corresponding to set {1, 0, 0.5}. In each case, both neurule bases had the same number of neurules. In this way, on the one hand, the symbolism-oriented and connectionism-oriented inference mechanisms and, on the other hand, the neurule-based explanation mechanism and the explanation mechanism of connectionist expert systems were compared in alternative schemes. It should be mentioned that the neurules constructed from the four datasets had negative as well as positive significance factors. The neurules constructed from the two symbolic rules bases had conditions with only positive significance factors.

7.1 Symbolism-oriented process vs connectionism-oriented process

An initial comment that can be made regarding the two inference processes is that the symbolism-oriented process is more natural than the connectionism-oriented process.

Due to the fact that the symbolism-oriented process performs inference in a similar way to 'classical' rule-based reasoning, inference steps are easily traceable. On the contrary, the inference steps of the connectionism-oriented process are more difficult to follow.

Table 2. Symbolism-oriented vs connectionism-oriented inference (set {1, -1, 0})

Datasets	*Connectionism-oriented process*		*Symbolism-oriented process*		
	Computations	*Asked Inputs*	*Computations*	*Asked Inputs*	*Mean Runtime Gain*
ACUTE (27, 6)	11.56	4.52	8.41	4.30	32.24%
LENSES (24, 4)	25.50	3.33	12.46	3.83	52.75%
CAR-DEP (1728, 6)	117.96	4.96	35.54	5.92	69.27%
NURSERY-DEP (12960, 8)	138.91	6.36	50.85	5.69	71.22%
CSRB	209.87	6.99	109.94	8	68.42%
MDRB	182.60	4.98	81.54	5	65.80%
AVERAGE	**114.40**	**5.19**	**49.79**	**5.46**	**59,95%**

Table 3. Symbolism-oriented vs connectionism-oriented inference (set {1, 0, 0.5})

Datasets	*Connectionism-oriented process (positive/negative sums)*		*Symbolism-oriented process (positive/negative sums)*		
	Computations	*Asked Inputs*	*Computations*	*Asked Inputs*	*Mean Runtime Gain*
ACUTE (27, 6)	13.15	5.19	9.22	4.59	40.37%
LENSES (24, 4)	26.46	3.58	11.46	3.88	61.10%
CAR-DEP (1728, 6)	168.40	6	43.52	5.94	77.19%
NURSERY-DEP (12960, 8)	180.40	6.50	49.74	6.78	75.74%
CSRB	194.37	7.84	84.92	7.93	70.54%
MDRB	165.16	6	71.31	5	64.38%
AVERAGE	**124.66**	**5.85**	**45.03**	**5.69**	**64.89%**

Experiments were conducted to compare the symbolism-oriented with the connectionism-oriented inference process. They were applied to the neurule bases produced from the above mentioned datasets and symbolic rule bases. The

© Elsevier

connectionism-oriented and symbolism-oriented processes corresponding to {1, 0, 0.5} set involve use of positive and negative remaining sums. The inference mechanisms are mainly compared as far as the computational cost for drawing conclusions is concerned. More specifically, comparison of the inference mechanisms is made in terms of the mean computational time and the mean number of computations required in drawing conclusions. Computations involve the mean number of times that a product of a significance factor and a corresponding condition value were added to the known sum of a neurule. Furthermore, we provide experiments involving the mean number of required (asked) input values for both inference mechanisms.

Tables 2 and 3 present experimental results regarding the performance of the inference mechanisms for the sets {1, -1, 0} and {1, 0, 0.5} respectively. “Mean Runtime Gain” represents the mean computational time gained by applying the symbolism-oriented inference mechanism compared to the connectionism-oriented one. “Computations” represents the mean number of computations required to reach conclusions. “Asked Inputs” represents the mean number of input (askable) values that were required to draw the conclusions. Beside each dataset name, two numbers are shown in parentheses: the total number of input combinations and the total (maximum possible) number of askable inputs in the domain. For instance, the expression “LENSES (24, 4)” means that for the LENSES dataset the total number of input combinations is 24 and the total number of askable inputs in the domain is 4. The inference mechanisms, except for the ACUTE dataset, were tested for all input combinations.

As shown in Tables 2 and 3, the symbolism-oriented process performs better than the connectionism-oriented process as far as runtime and mean number of computations are concerned. More specifically, symbolism-oriented process results in a 56.47% average reduction in the number of computations (49.79 vs 114.40) for the {1, -1, 0} value set and in a 63.88% reduction (45.03 vs 124.66) for the {1, 0, 0.5} value set. Also, it provides a 59.95% mean runtime gain for the {1, -1, 0} value set and a 64.89% for the {1, 0, 0.5} value set. The main reason for this is the fact that in the connectionism-oriented process, evaluation of a condition in any inference affects several neurules and all its related conditions. On the contrary, in the symbolism-oriented process, each neurule is evaluated separately and some computations performed in the connectionism-oriented process are spared. Due to the above, in the connectionism-oriented process the number of computations and thus the time required to perform inference are worse than the ones in the symbolism-oriented process. It should also be mentioned that there is no clear advantage of either inference process as far as the mean number of asked inputs is concerned. Furthermore, there is no clear advantage of either inference process as far as the set of input values {1, -1, 0} or {1, 0, 0.5} for inputs is concerned. However, in the case of neurule bases having conditions with only positive significance factors, as in the case of neurule bases produced from conversion of symbolic rule bases, both inference processes perform better for the set {1, 0, 0.5} compared to the set of discrete values {1, -1, 0}. Therefore, when converting symbolic rule bases to neurule bases, one could examine the alternative neurule bases corresponding to both sets of input values.

7.2 Alternative schemes for symbolism-oriented process

Experiments were conducted to evaluate the performance of the symbolism-oriented process using positive and negative sums compared to the symbolism-oriented

process using only plain remaining sum as well as the symbolism-oriented process according to (Hatzilygeroudis & Prentzas, 2000) for the set of input values {1, 0, 0.5}. Table 4 presents results of those experiments (results for symbolism-oriented process with positive and negative sums are replicated from Table 3). "Comp." represents the mean number of computations required to reach conclusions as in previous tables. "VR" represents the mean number of rules visited during inference.

Table 4. Results for the three alternative symbolism-oriented processes

Datasets	*Symbolism-oriented process (plain remaining sum)*			*Symbolism-oriented process (positive/negative sums)*			*Symbolism-oriented process (Hatzilygeroudis & Prentzas 2000)*		
	Comp.	*VR*	*Asked Inputs*	*Comp.*	*VR*	*Asked Inputs*	*Comp.*	*VR*	*Asked Inputs*
ACUTE (27, 6)	10.93	5.15	5.26	9.22	4.70	4.59	12.96	5.33	5.63
LENSES (24, 4)	15.04	2.67	4	11.46	2.67	3.88	19	2.67	4
CAR-DEP (1728, 6)	64.29	11.44	5.94	43.52	11.17	5.94	81.17	11.88	6
NURSERY-DEP (12960, 8)	76.77	15.04	6.91	49.74	13.70	6.78	119.96	17.64	8
CSRB	86.99	39.43	7.96	84.92	39.43	7.93	84.92	39.43	7.93
MDRB	71.73	33.88	5	71.31	33.88	5	71.31	33.88	5
AVERAGE	**54.29**	**17.94**	**5.85**	**45.03**	**17.59**	**5,69**	**64,89**	**18.47**	**6.09**

Table 5. Mean runtime gain of symbolism-oriented process with positive and negative remaining sums compared to the other two symbolism-oriented processes

Datasets	*Symbolism-oriented process (positive/negative sums) vs Symbolism-oriented process (plain remaining sum)*	*Symbolism-oriented process (positive/negative sums) vs Symbolism-oriented process (Hatzilygeroudis & Prentzas 2000)*
ACUTE (27, 6)	8.45%	14.47%
LENSES (24, 4)	14.28%	22.20%
CAR-DEP (1728, 6)	16.50%	7.20%
NURSERY-DEP (12960, 8)	17.35%	33.06%
CSRB	13.16%	-
MDRB	1.89%	-
AVERAGE	**11.94%**	**19.23%**

In case positive and negative sums are used, performance is better as far as runtime, computations, visited rules and asked inputs are concerned. These results were expected since the positive and negative sums contribute in evaluating neurules with fewer of their conditions needing to be evaluated. Fewer evaluated conditions means that time-performance is improved (in terms of runtime, computations and visited rules) and also fewer inputs need to be asked. The performance of the

symbolism-oriented process with positive and negative sums is identical to the performance of the process according to (Hatzilygeroudis & Prentzas, 2000) in case of neurule bases with only positive significance factors. In case of neurule bases with both positive and negative significance factors, the performance of the process according to (Hatzilygeroudis & Prentzas, 2000) is worse than the performance of the symbolism-oriented process using positive and negative sums as well as the symbolism-oriented process using only plain remaining sum. In case of neurule bases with only positive significance factors, the performance of the process according to (Hatzilygeroudis & Prentzas, 2000) is better than the performance of the symbolism-oriented process using only plain remaining sum.

Table 5 presents results involving the mean runtime gain of the symbolism-oriented process with positive and negative remaining sums compared to the symbolism-oriented process with plain remaining sum and the symbolism-oriented process according to (Hatzilygeroudis & Prentzas, 2000).

7.3 'How' Explanation process in neurules and connectionist expert systems

The 'how' explanation process associated with neurules compared to the one used in connectionist expert systems displays the following advantages:

a) It produces more natural explanation rules. This is due to the fact that the random cells introduced in connectionist knowledge bases have no meaning to the user. These random cells downgrade the naturalness of the explanation rules since the "concepts" corresponding to the random cells appear in conditions and conclusions of the explanation rules.

b) It produces less explanation rules. Once again this is due to the existence of random cells in connectionist knowledge bases. The increase in the number of explanation rules in connectionist expert systems increases the computational cost in producing the explanation rules. Furthermore, it raises an issue of comprehensibility as far as the user is concerned. The more explanation rules are presented to the user, the more confused he/she will be.

We use an example to demonstrate this. Table 6 depicts in the form of a matrix the connectionist knowledge base corresponding to the neurule base presented in Table 1. The first column of the table depicts the names of input and intermediate nodes whereas the first row depicts the names of intermediate and output nodes. Fig. 2 graphically depicts the connectionist network. The connectionist knowledge base has been constructed using the dependency information concerning the input, intermediate and output variables. Training is performed separately for each intermediate and output node and the corresponding inputs. The training sets have been extracted from the truth tables of the combined logical function of the rules of the merger set corresponding to each intermediate and output conclusion. The connectionist knowledge base contains five random cells, two among the 'arthritis' output node and its inputs and three among the 'primary-malignant' output node and its inputs. According to the instructions given by Gallant (1993), input nodes are connected to both random cells and the output nodes. The random cells have been inserted in order to handle inseparability and have no meaning to the user as they do not correspond to domain variables.

The right part of Table 7 contains the explanation rules produced from the connectionist knowledge base for the inference example outlined in Section 6. The left part of the table contains the explanation rules produced by the explanation mechanism associated with neurules. Variables 'uD' and 'uE' correspond to the random cells introduced into the connectionist knowledge base in order to deal with

inseparability. The existence of meaningless intermediate variables, in the connectionist knowledge base, increases the number of its explanation rules and makes them difficult to comprehend since variables ‘uD’ and ‘uE’ make no sense to the user.

Table 6. The connectionist knowledge base corresponding to the neurule base shown in Table 1

	human0-20	human21-35	human36-55	Int. var1 arthritis (uA)	Int. var2 arthritis (uB)	Int. var1 primary-malignant (uC)	Int. var2 primary-malignant (uD)	Int. var3 primary-malignant (uE)	arthritis	primary-malignant
bias	-0.5	-2.2	-4.1	10.3	13.9	10.3	-50.9	3.1	4.9	10.3
man	3.4	1.5	2.6	0	0	0	0	0	0	0
woman	3.0	1.6	3.2	0	0	0	0	0	0	0
age<=20	3.3	0	0	0	0	0	0	0	0	0
age>20	0	1.3	0	0	0	0	0	0	0	0
age<=35	0	1.5	0	0	0	0	0	0	0	0
age>35	0	0	3.0	0	0	0	0	0	0	0
age<=55	0	0	3.3	0	0	0	0	0	0	0
pain-night	0	0	0	-15.3	-0.9	-18.9	-4.5	-11.7	0.9	20.7
pain-continuous	0	0	0	-11.1	-7.5	6.9	-3.9	-11.1	5.1	10.5
ant-reaction-none	0	0	0	0	0	-11.0	7.0	-18.2	0	14.2
ant-reaction-medium	0	0	0	-14.6	-0.2	0	0	0	1.6	0
ant-reaction-high	0	0	0	-0.8	-4.4	0	0	0	1.0	0
fever-no-fever	0	0	0	-7.8	-0.6	13.8	-4.2	-15.0	1.2	10.2
fever-low	0	0	0	0	0	-4.3	-11.5	2.9	0	10.1
fever-medium	0	0	0	0	0	-1.0	-11.8	2.6	0	9.8
fever-high	0	0	0	0	0	-3.9	-11.1	6.9	0	10.5
joints-pain-no	0	0	0	-0.3	-3.9	0	0	0	1.5	0
human0-20	0	0	0	-11.0	-3.8	-14.6	-14.6	7.0	-2.0	7.0
human21-35	0	0	0	-15.1	-4.3	-0.8	-26.0	10.0	1.1	6.4
human36-55	0	0	0	-1.0	-8.2	-7.4	-0.2	-25.4	0.8	-0.2
Int. var1 arthritis (uA)	0	0	0	0	0	0	0	0	-12.8	0
Int. var2 arthritis (uB)	0	0	0	0	0	0	0	0	-6.2	0
Int. var1 primary-malignant (uC)	0	0	0	0	0	0	0	0	0	-11.6
Int. var2 primary-malignant (uD)	0	0	0	0	0	0	0	0	0	-21.9
Int. var3 primary-malignant (uE)	0	0	0	0	0	0	0	0	0	-18.4

© Elsevier

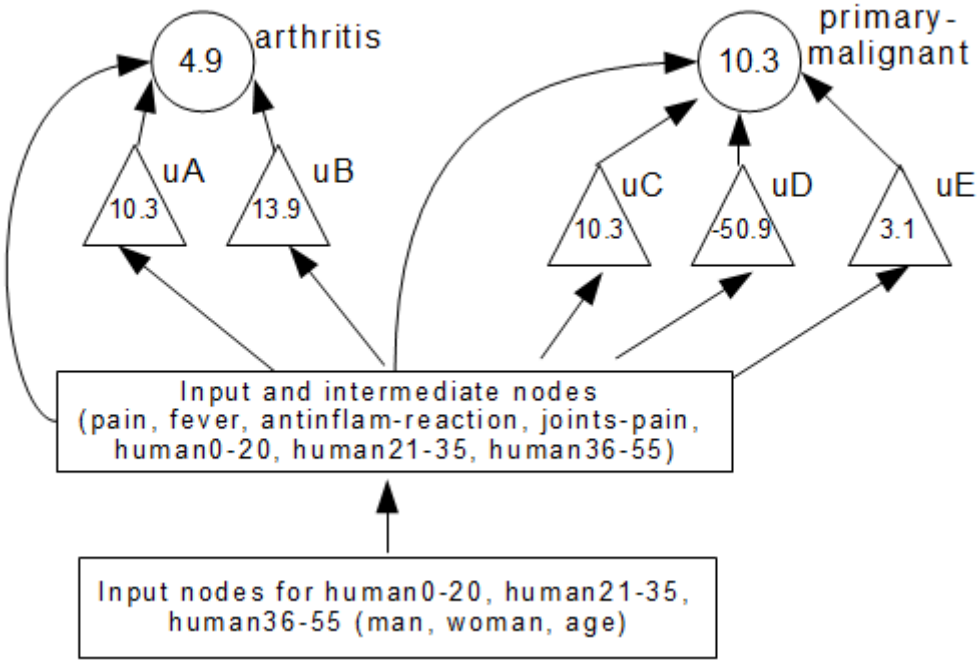

Fig. 2. Graphical depiction of the connectionist knowledge base

Table 7. Example 'how' explanation rules produced from neurules and connectionist expert systems

NEURULES	CONNECTIONIST EXPERT SYSTEMS	
EXR1 **if** gender is man and age <= 35 and age > 20 **then** patient-class is human21-35	GH-EXR1 **if** sex is man and age <= 35 and age > 20 **then** patient-class is human21-35	GH-EXR3 **if** antinflam-reaction is none and fever is no-fever and pain is night-pain, and patient-class is human21-35 **then** uE isnot true
EXR2 **if** antinflam-reaction is none and pain is night-pain and patient-class is human21-35 **then** disease-type is primary-malignant	GH-EXR2 **if** patient-class is human21-35 **then** uD isnot true	GH-EXR4 **if** uD isnot true and pain is night-pain and uE isnot true and antinflam-reaction is none **then** disease-type is primary-malignant

Experiments were conducted to compare the 'how' explanation process associated with neurules with the 'how' explanation process used in connectionist expert systems. In the experiments involving connectionist expert systems, we used the Recency Inference Engine presented in (Ghalwash, 1998) which is more efficient than MACIE presented in (Gallant, 1993).

Tables 8 and 9 present some experimental results comparing the 'how' explanation process associated with neurules with the 'how' explanation process used in connectionist expert systems. The explanations produced are the ones justifying the conclusions reached by the corresponding inference mechanisms for the six knowledge bases when giving the same inputs as in the previous section. Comparison is made in terms of the number of explanation rules produced and the time required producing them. Column 'EXRUL-NUM' contains the mean number of explanation rules produced. Column 'Mean Runtime Gain' represents the mean computational

© Elsevier

time gained by applying the 'how' explanation process associated with neurules with the 'how' explanation process used in connectionist expert systems.

Table 8. 'How' type explanations: neurules vs. connectionist expert systems (set of discrete values {1, -1, 0} for inputs)

Datasets	*Neurules (Symbolism-oriented process)*		*Connectionist expert systems*
	EXRUL-NUM	*Mean Runtime Gain*	*EXRUL-NUM*
ACUTE (27)	3.07	40%	4.30
LENSES (24)	1	42.10%	2.42
CAR-DEP (1728)	4.10	55.17%	7.56
NURSERY-DEP (12960)	3.71	54.90%	7.55
CSRB	2.74	89.09%	18.90
MDRB	1	58.62%	3.11
AVERAGE	**2.60**	**56.65%**	**7.31**

As can be seen from the tables, the number of 'how' explanation rules produced by the explanation mechanism associated with neurules is less than the number of explanation rules produced by the explanation mechanism used in connectionist expert systems. For this reason, the time required to produce explanation rules from neurules is less than the time required to produce explanation rules in connectionist expert systems. More specifically, symbolism-oriented process results in a 56.65% average mean runtime gain for the {1, -1, 0} value set and 56.73% for the {1, 0, 0.5} value set. Also, it results in an average reduction in the number of produced explanation rules of 64.38% (2.6 vs 7.3) for the {1, -1, 0} value set and of 69.28% (2.8 vs 9.11) for the {1, 0, 0.5} value set.

Table 9. 'How' type explanations: neurules vs. connectionist expert systems (set of discrete values {1, 0, 0.5} for inputs)

Datasets	*Neurules (Symbolism-oriented process)*		*Connectionist expert systems*
	EXRUL-NUM	*Mean Runtime Gain*	*EXRUL-NUM*
ACUTE (27)	3.04	21.42%	5.37
LENSES (24)	1	73.33%	3.46
CAR-DEP (1728)	4.35	51.65%	9.77
NURSERY-DEP (12960)	4.83	52.17%	10.63
CSRB	2.62	88.14%	22.08
MDRB	1	53.66%	3.37
AVERAGE	**2.80**	**56.73%**	**9.11**

8. Conclusions

In this paper, we present inference and explanation mechanisms for neurules, a type of hybrid rules integrating symbolic rules with neurocomputing. Each neurule is considered as an adaline neural unit expressed in symbolic oriented syntax. An attractive feature of neurules is that compared to other similar neuro-symbolic approaches, like the one followed in connectionist expert systems, they retain the modularity and to some degree the naturalness of symbolic rules; also they support interactivity with the user during inference by asking for input values when necessary. Neurules are constructed either by converting existing symbolic rules or directly from empirical data in the form of training examples.

We present two alternative neurule-based inference processes. The first, called connectionism-oriented, gives pre-eminence to and is based on neurocomputing. Its main idea was presented in a past work of ours. Here, we present an improved version. The second presented inference process, called symbolism-oriented process,

gives pre-eminence to symbolic reasoning and is based on a backwards chaining like approach. For each inference process, we also provide schemes for two alternative sets of discrete input values, {1, -1, 0} and {1, 0, 0.5}. Inference is more natural when the set {1, 0, 0.5} is used. Experimental results have shown that the symbolism-oriented inference process requires less computational effort to draw conclusions compared to the connectionism-oriented inference process, for both alternative input values sets. More specifically, symbolism-oriented process results in a 56.47% (resp. 63.88%) average reduction in the number of computations for the {1, -1, 0} (resp. {1, 0, 0.5}) value set. Also, it provides a 59.95% (resp. 64.89%) mean runtime gain for the {1, -1, 0} (resp. {1, 0, 0.5}) value set.

The explanation mechanism provides three types of explanations: 'how', 'why' and 'why-not'. The 'how' explanation rules, produced by the neurule-based explanation mechanism are more natural and less in number than the ones produced by the corresponding mechanism in connectionist expert systems. More specifically, symbolism-oriented process results in a 56.65% (resp. 56.73%) average mean runtime gain for the {1, -1, 0} (resp. {1, 0, 0.5}) value set. Also, it results in an average reduction in the number of produced explanation rules of 64.38% (resp. 69.28%) for the {1, -1, 0} (resp. {1, 0, 0.5}) value set.

Our future work involves various research directions. We outline the most important in the following.

A first research direction concerns the combination of neurule-based with case-based reasoning. A number of approaches have been developed that combine rules and cases. This type of combination provides advantages due to the fact that rule-based and case-based reasoning have complementary advantages and disadvantages as far as knowledge representation and reasoning are concerned (Prentzas & Hatzilygeroudis, 2007). As neurules retain the naturalness and modularity of symbolic rules, the combination of neurules and cases provides advantages. In (Prentzas, Hatzilygeroudis, & Michail, 2008) we present an approach combining neurule-based and case-based reasoning that improves a previous approach of ours in terms of accuracy. We intend to apply this approach to a financial domain. Neurules will be produced from available symbolic rules. A prerequisite for the application of an approach combining rule-based and case-based reasoning is the availability of rules and cases. This prerequisite is satisfied in various financial domains.

A second direction involves testing of different similarity measures in the construction process of neurules from training examples. Neurules, as mentioned, are based on the adaline unit. In the construction process, the inherent difficulty of the adaline unit to handle inseparable training sets is dealt with by splitting inseparable training sets into subsets, which are trained separately. The splitting process creates subsets of examples which are 'close' to each other. An aspect of interest is how the application of different similarity measures affects the splitting process in terms of the number of constructed neurules. In a previous work, we presented initial results involving four different similarity measures. We intend to perform further experiments with other similarity measures for categorical data such as the ones brought together from different fields in (dos Santos & Zárate, 2015) and compared using a common clustering approach. We intend to test and compare corresponding similarity measures in the construction process of neurules. As similarity measures are used in various domains, their comparative testing may yield interesting results for the research community.

Another research direction in the construction process of neurules from training examples concerns the methods used for creation of training subsets. The splitting

process resembles clustering approaches (Kaufman & Rousseeuw, 2005; Aggarwal & Reddy, 2014) as the goal is to create subsets consisting of 'close' examples. Clustering approaches for categorical data have gained interest during the last years. We intend to test alternative clustering approaches in the creation of training subsets. An example of a clustering approach that could be applied is k-medoids (Park & Jun, 2009).

Yet another direction is the application of neurules to an e-learning system addressed to early childhood. Early childhood education is a field gathering the interest of educational technology researchers. The number of software applications addressed to early childhood has increased significantly in the recent years (Morgan & Siraj-Blatchford, 2013). Few e-learning systems incorporating AI methods and addressed to early childhood have been developed (Prentzas, 2013). Hybrid AI approaches may provide advantages, as they satisfy the knowledge representation requirements for intelligent tutoring systems (ITSs) (Hatzilygeroudis & Prentzas, 2006). We intend to extend our previous work by developing ITSs that incorporate neurules (Prentzas, Hatzilygeroudis, & Koutsojannis, 2001; Hatzilygeroudis & Prentzas, 2004b) related to early childhood.

A final interesting research direction is the combination of fuzzy logic with neurules. In (Koutsojannis & Hatzilygeroudis, 2006) a type of fuzzy neurules based on fuzzy adaline units is presented. Reasoning and explanation in such an integrated representation scheme have not yet been explored.

References

Aggarwal, C. C., & Reddy, C. K. (Eds.) (2014). *Data clustering: algorithms and applications*. Boca Raton, FL: CRC Press.

Bache K., & Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Bohanec, M., & Zupan, B. (1997). AI Lab Datasets [http://magix.fri.uni-lj.si/blaz/hint/datasets.htm]. Slovenia, University of Ljubljana, Faculty of Computer and Information Science.

Chuang, C.-L., & Huang, S.-T. (2011). A hybrid neural network approach for credit scoring. *Expert Systems, 28,* 185–196.

Evans, M., & Kennedy, J. (2014). Integration of Adaptive Neuro Fuzzy Inference Systems and principal component analysis for the control of tertiary scale formation on tinplate at a hot mill. *Expert Systems with Applications, 41,* 6662–6675.

Gallant, S.I. (1993). *Neural network learning and expert systems*. Boston, MA: MIT Press.

Garcez, D'Avila, A. S., Broda, K., & Gabbay, D. M. (2002). *Neural-symbolic learning systems: foundations and applications, Perspectives in Neural Computing*. Berlin/Heidelberg: Springer-Verlag.

Garcez, D'Avila, A. S., Lamb, L. C., & Gabbay, D. M. (2009). *Neural-symbolic Cognitive Reasoning*. Berlin/Heidelberg: Springer-Verlag.

Garcez, D'Avila, A. S., & Lamb, L. C. (2011). Cognitive algorithms and systems: reasoning and knowledge representation. In V. Cutsuridis, A. Hussain, & J. G. Taylor (Eds.), *Perception-action Cycle: Models, Architectures, and Hardware, Springer*

Series in Cognitive and Neural Systems (pp. 573–600). Berlin/Heidelberg: Springer-Verlag.

Ghalwash, A. Z. (1998). A recency inference engine for connectionist knowledge bases. *Applied Intelligence, 9,* 201–215.

Hatzilygeroudis, I., Vassilakos, P. J. & Tsakalidis, A. (1997). XBONE: A hybrid expert system for supporting diagnosis of bone diseases. In C. Pappas, N. Maglaveras, & J.-R. Scherrer (Eds.), *Proceedings of the Medical Informatics Europe'97 (MIE'97)* (pp. 295-299). Amsterdam: IOS Press.

Hatzilygeroudis, I., & Prentzas, J. (2000). Neurules: improving the performance of symbolic rules. *International Journal on Artificial Intelligence Tools, 9,* 113–130.

Hatzilygeroudis, I., & Prentzas, J. (2001). Constructing modular hybrid knowledge bases for expert systems. *International Journal on Artificial Intelligence Tools, 10,* 87–105.

Hatzilygeroudis, I., & Prentzas, J. (2004a). Neuro-symbolic approaches for knowledge representation in expert systems. *International Journal of Hybrid Intelligent Systems, 1,* 111–126.

Hatzilygeroudis, I., & Prentzas, J. (2004b). Using a hybrid rule-based approach in developing an intelligent tutoring system with knowledge acquisition and update capabilities. *Expert Systems with Applications, 26,* 477–492.

Hatzilygeroudis, I., & Prentzas, J. (2006). Knowledge representation in Intelligent Educational Systems. In Z. Ma (Ed.), *Web-based intelligent e-learning systems: technologies and applications* (pp. 175-192). Hershey, PA: Information Science Publishing.

Hatzilygeroudis, I., & Prentzas, J. (2010). Integrated rule-based learning and inference. *IEEE Transactions on Knowledge and Data Engineering, 22,* 1549–1562.

Hatzilygeroudis, I., & Prentzas, J. (Eds.) (2011a). *Combinations of intelligent methods and applications, Smart Innovation, Systems and Technologies,* vol. 8. Berlin/Heidelberg: Springer-Verlag.

Hatzilygeroudis, I., & Prentzas, J. (2011b). Fuzzy and neuro-symbolic approaches to assessment of bank loan applicants. In L. Iliadis, I. Maglogiannis, & H. Papadopoulos (Eds.), *Proceedings of the 12th Engineering Applications of Neural Networks SIG International Conference and 7th Artificial Intelligence Applications and Innovations IFIP WG 12.5 International Conference. IFIP Advances in Information and Communication Technology (AICT). Vol. 36* (pp. 82–91). Berlin/Heidelberg: Springer-Verlag.

Holldobler, S., & Kalinke, Y. (1994). Towards a massively parallel computational model for logic programming. In *Workshop on Combining Symbolic and Connectionist Processing, ECCAI'94* (pp. 68–77). Amsterdam.

Huang, H.-X., Li, J.-C., & Xiao, C.-L. (2015). A proposed iteration optimization approach integrating backpropagation neural network with genetic algorithm. *Expert Systems with Applications, 42,* 146–155.

Kaufman, L., & Rousseeuw, P. J. (2005). *Finding groups in data: an introduction to cluster analysis.* New York, NY: John Wiley & Sons.

Koutsojannis, C., & Hatzilygeroudis, I. (2006). FUNEUS: A neurofuzzy approach based on fuzzy adaline neurons. In L. Ponserini, P. Peppas and A. Perini (Eds.), *Proceedings of Third Starting AI Researchers' Symposium. Frontiers in Artificial Intelligence and Applications. Vol. 142* (pp. 96–107). Amsterdam: IOS Press.

Lin, X., Sun, J., Palade, V., Fang, W., Wu, X., & Xu, W. (2012). Training ANFIS parameters with a quantum-behaved particle swarm optimization algorithm. In Y. Tan, Y. Shi, & Z. Ji (Eds.), *Proceedings of the Third International Conference on Swarm Intelligence. Lecture Notes in Computer Science. Vol. 7331* (pp. 148–155). Berlin/Heidelberg: Springer-Verlag.

Morgan, A., & Siraj-Blatchford, J. (2013). *Using ICT in the early years: parents and practitioners in partnership.* London: MA Education.

Park, H. S., & Jun, C. H. (2009). A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications, 36,* 3336–3341.

Prentzas, J., Hatzilygeroudis, I., & Koutsojannis, C. (2001). A Web-based ITS controlled by a hybrid expert system. In T. Okamoto, R. Hartley, Kinshuk, J. P. Klus (Eds.), *Proceedings of the IEEE International Conference on Advanced Learning Technologies* (pp 239–240). Los Alamitos, CA: IEEE Computer Society.

Prentzas, J., & Hatzilygeroudis, I. (2007). Categorizing approaches combining rule-based and case-based reasoning. *Expert Systems, 24,* 97–122.

Prentzas, J., Hatzilygeroudis, I., & Michail, O. (2008). Improving the integration of neuro-symbolic rules with case-based reasoning. In J. Darzentas, G. A. Vouros, S. Vosinakis, & A. Arnellos (Eds.), *Proceedings of the Fifth Hellenic Conference on AI. Lecture Notes in Computer Science. Vol. 5138* (pp 377–382). Berlin/Heidelberg: Springer-Verlag.

Prentzas, J., & Hatzilygeroudis, I. (2009). Combinations of case-based reasoning with other intelligent methods. *International Journal of Hybrid Intelligent Systems, 6,* 189–209.

Prentzas, J., & Hatzilygeroudis, I. (2011). Neurules – a type of neuro-symbolic rules: an overview. In I. Hatzilygeroudis, & J. Prentzas (Eds.), *Combinations of Intelligent Methods and Applications. Smart Innovation, Systems and Technologies. Vol. 8* (pp. 145–165). Berlin/Heidelberg: Springer-Verlag.

Prentzas, J. (2013). Artificial Intelligence methods in early childhood education. In X.-S. Yang (Ed.), *Artificial Intelligence, evolutionary computation and metaheuristics – In the footsteps of Alan Turing. Studies in Computational Intelligence. Vol. 427* (pp. 169–199). Berlin/Heidelberg: Springer-Verlag.

dos Santos, T. R. L., & Zárate, L. E. (2015). Categorical data clustering: what similarity measure to recommend?. *Expert Systems with Applications, 42,* 1247–1260.

Towell, G., & Shavlik, J. (1994). Knowledge-based artificial neural networks. *Artificial Intelligence, 70,* 119–165.

Tweedale, J. W., & Jain, L. C. (Eds.) (2014). *Recent advances in knowledge-based paradigms and applications: enhanced applications using hybrid Artificial Intelligence techniques. Advances in Intelligent Systems and Computing. Vol. 234*. Berlin/Heidelberg: Springer-Verlag.

Zhang, W., Ma, H., & Yang, S. X. (2015). A neuro-fuzzy decoupling approach for real-time drying room control in meat manufacturing. *Expert Systems with Applications, 42,* 1039–1049.

© Elsevier