Published in Expert Systems, Vol. 32, No. 2, pp. 244-260, 2015.

Improving Efficiency of Merging Symbolic Rules into Integrated Rules: Splitting Methods and Mergability Criteria

Jim Prentzas[1], Ioannis Hatzilygeroudis[2]

(1) Democritus University of Thrace, School of Education Sciences
Department of Education Sciences in Early Childhood, Laboratory of Informatics
68100 Nea Chili, Alexandroupolis, Greece
Email: dprentza@psed.duth.gr

(2) University of Patras, School of Engineering
Department of Computer Engineering & Informatics
26500 Patras, Greece
Email: ihatz@ceid.upatras.gr

Abstract: *Neurules are a type of neuro-symbolic rules integrating neurocomputing and production rules. Each neurule is represented as an adaline unit. Neurules exhibit characteristics such as modularity, naturalness and ability to perform interactive and integrated inferences and provide explanations for reached conclusions. One way of producing a neurule base is through conversion of an existing symbolic rule base yielding an equivalent but more compact rule base. The conversion process merges symbolic rules having the same conclusion into one or more neurules. Due to the inability of the adaline unit to handle inseparability, more than one neurule for each conclusion may be produced by splitting the initial set of symbolic rules into subsets. This paper presents research work improving the conversion process in terms of runtime and number of produced neurules. First, we show how easier is to construct a neurule base than a connectionist one. Second, we present alternative rule set splitting methods. Finally, we define criteria concerning the ability or inability to convert a rule set into a single, equivalent, but more compact rule. With application of such mergability criteria, the conversion process of symbolic rules into neurules becomes more time-efficient. All the above are supported by experimental results.*

1. Introduction

A research direction in Artificial Intelligence (AI) involves the integration (or combination) of two or more intelligent methods. The purpose of such integrations is to create improved approaches and systems in terms of knowledge representation and reasoning. This is achieved by exploiting the corresponding advantages of the component approaches and surpassing their disadvantages. Research results concerning various application domains have demonstrated that integrated intelligent approaches may yield advantages (Abraham et al. 2012; Castillo et al. 2012; Hatzilygeroudis & Prentzas 2011a and b; Sahin et al. 2012). Successful integrations usually combine intelligent approaches with complementary advantages and disadvantages. Several types of integrated approaches have been proposed. A large portion of integrations combines soft computing methods (e.g. fuzzy logic, neural networks and genetic algorithms) among themselves or with other AI methods such as logic, rules and machine learning. Some types of combinations have been extensively used and explored. Examples of such combinations, among others, involve neuro-symbolic approaches integrating neural networks with symbolic methods (Garcez & Lamb 2011; Hatzilygeroudis & Prentzas 2004), neuro-fuzzy approaches integrating neural networks with fuzzy methods (Chattopadhyay 2013; Sreekantha & Kulkarni 2012; Lin et al. 2012), approaches combining neural networks and genetic algorithms (Belciug & Gorunescu 2013) and approaches combining case-based reasoning with rule-based reasoning (Prentzas & Hatzilygeroudis 2007) or other intelligent methods (Chuang & Huang 2011).

Different types of integrations combining neural and symbolic approaches have been presented (Garcez et al. 2002; Garcez et al. 2009). A large number of neuro-symbolic approaches combine neural networks with symbolic rules. Neural networks and symbolic rules have complementary advantages and disadvantages (Hatzilygeroudis & Prentzas 2004) and their combination constitutes a popular research trend. Advantages of symbolic rules concern naturalness, modularity and availability of interactive inference mechanism as well as explanation capabilities. Their main disadvantages involve difficulties in knowledge acquisition as well as in reasoning with unknown inputs. Neural networks, on the other hand, are constructed from existing training examples and are able to generalize well. However, they lack characteristics of rules such as naturalness, modularity and ability to provide explanations for reached outputs.

In most of the neuro-symbolic approaches combining symbolic rules with neurocomputing that result into a seamless combination of the integrated components, the main component is the neural one. More specifically, the symbolic approach is incorporated in or mapped to the neural component. One may discern two primary trends in this research direction.

The one trend primarily focuses on modeling symbolic processes in a neural way. This trend stems from (Holldobler & Kalinke 1994) where a connectionist network is developed that implements the meaning function of a propositional (definite) logic program. Following this line of research, more advanced systems have been proposed that extend that approach to first-order logic and multivalued logic-based programs as well as nonclassical logics. The other trend stems from (Towell & Shavlik 1994), where a background domain theory in the form of propositional rules is used to construct an initial neural network, which is then finally trained through a training data set. This leads to theory refinement. Furthermore, connectionist expert systems are considered as integrated systems that represent relationships between concepts associated with nodes in a neural network (Gallant 1993; Ghalwash 1998). Connectionist expert systems provide inference and explanation mechanisms but explanation rules include meaningless conditions since certain network nodes do not correspond to domain concepts.

A common aspect of the aforementioned approaches is that they give pre-eminence to the neural component and consequently, to a large degree, they lack advantages of symbolic rules. In contrast to such approaches, neurules (Hatzilygeroudis & Prentzas 2010, Prentzas & Hatzilygeroudis 2011) give preeminence to the symbolic component. Neurules are a type of integrated rules seamlessly combining symbolic rules (of propositional type) and neurocomputing (adaline approach). They retain the naturalness and modularity of symbolic rules in a large degree. Also a neurule-based system possesses an interactive inference mechanism (Hatzilygeroudis & Prentzas 2010) and an explanation mechanism to provide explanations for drawn conclusions (Prentzas & Hatzilygeroudis 2012b). Neurules can be produced either from symbolic rules (Hatzilygeroudis & Prentzas 2000) or from empirical data (Hatzilygeroudis & Prentzas 2001).

A neurule base can be produced from a symbolic rule base by applying a conversion process (Hatzilygeroudis & Prentzas 2000). Conversion does not involve refinement of the symbolic rule base, but creates an equivalent knowledge base. This means that the conclusions drawn from the neurule base are the same as those drawn from the symbolic rule base, given the same inputs. Each produced neurule usually merges two or more symbolic rules with the same conclusion. Therefore, the size of the produced neurule base is less than that of the symbolic rule base as far as both the number of

rules and the number of conditions are concerned. This results in improvements to the efficiency of the inferences from the neurule base, compared to those from the symbolic rule base, as shown in (Hatzilygeroudis & Prentzas 2000).

The conversion process tries to merge all symbolic rules having the same conclusion into a single neurule. However, this is not always possible, due to the inability of the adaline unit to handle inseparability, and thus more than one neurule for each conclusion may be produced. So, in cases of inseparability, the initial set of rules is split into two disjoint subsets that may lead to separability. Splitting is based on rule *closeness*: each subset contains rules that are close to each other. Closeness is based on the number of identical conditions between rules (introduced in Hatzilygeroudis & Prentzas 2000).

There are two directions that the conversion process has not been explored for possible improvements. The one concerns exploration of employing splitting strategies not based on the simple closeness criterion (above mentioned), but on a more complicated. Given that splitting of an initial set of rules having the same conclusion into disjoint subsets of close rules resembles clustering approaches, applying a clustering algorithm to rule splitting may lead to better results. The other direction concerns definition of mergability criteria, i.e. determining whether a set of rules can (or cannot) be merged into a single neurule, without applying training. This would save a number of useless trainings.

So, in this paper, first, we use data derived from available symbolic rule bases to construct knowledge bases utilized in connectionist expert systems, which we compare with equivalent neurule bases, constructed by merging symbolic rules. The comparison demonstrates the advantages of neurules in terms of representation naturalness and ease of construction process. Second, we present new splitting methods used in the conversion process in case of inseparability, which are based on the notion of clustering. One of them does better than the initial one in certain cases. Finally, we (re)define and revise criteria (initially introduced in Prentzas & Hatzilygeroudis 2012a) concerning the ability or inability to convert a rule set into a single neurule. With application of such criteria, the conversion process of symbolic rules to neurules becomes more efficient by avoiding trainings not directly producing neurules. The definition of such criteria is of general representational interest too.

This paper is organized as follows. Section 2 presents work related to connectionist expert systems and some kind of rule merging. Section 3 outlines main aspects of neurules. Section 4 presents the conversion process and provides relevant examples. Section 5 presents alternative splitting methods that may be incorporated in the conversion process. Section 6 discusses criteria for efficiently merging symbolic rules into neurules. Section 7 presents experimental results derived from applying the conversion process to available symbolic rule bases. Finally, Section 8 concludes.

2. Related Work

In this section, we present and discuss work related to two aspects. One aspect concerns connectionist expert system bases and the other approaches regarding rule merging.

2.1 Connectionist Expert Systems

Connectionist expert systems constitute an approach providing inference and explanation mechanisms (Gallant 1993; Ghalwash 1998). The knowledge base in connectionist expert systems is constructed using dependency information (i.e.

associations between input, intermediate and output concepts) and training examples. A connectionist knowledge base could be constructed from a set of symbolic rules. This could be done by using: (a) the dependency information represented by rules and (b) the training sets extracted from the truth table of the combined logical function of each initial merger (see Section 4). Compared to neurules, connectionist expert systems have certain disadvantages involving knowledge representation naturalness, knowledge base construction process, provision of explanations and inference process. These disadvantages are discussed in the following.

The knowledge bases of connectionist expert systems lack the naturalness of neurules. Domain concepts in connectionist expert systems are associated with network nodes and relationships between concepts are denoted by node connections. However, the network also includes nodes that are not associated with domain concepts. These nodes (called 'random cells') are randomly generated to represent non-separable examples. Representation is meaningless in the parts of the network involving random cells. In (Hatzilygeroudis & Prentzas 2001) illustrative examples are provided involving the creation of a neurule and a connectionist knowledge base from a set of training examples showing the advantages of neurules.

The construction process of a neurule base is straightforward. On the contrary, the construction of the knowledge base in a connectionist expert system is not straightforward and requires more effort. There is no specific way to determine the mapping that random cells will implement as well as the number of the required random cells. This is illustrated in this paper.

Connectionist expert systems provide explanations about how outputs-conclusions are reached. Explanation is given in the form of if-then rules (Gallant 1993). Conditions and conclusions of explanation rules however include variables pertaining to random cells and are therefore meaningless. The neurule-based explanation mechanism is discussed in (Prentzas & Hatzilygeroudis 2012b). Explanation rules provided by the neurule-based explanation mechanism are more natural since their conditions and conclusions involve only domain concepts. Experimental results also show that the neurule-based explanation mechanism provides less explanation rules compared to the corresponding explanation mechanism used in connectionist expert systems. Explanation aspects are not discussed in this paper.

In (Hatzilygeroudis & Prentzas 2010) the neurule-based inference mechanism is compared with the inference mechanism used in connectionist expert systems. Experimental results have shown that the neurule-based inference is more efficient. The neurule bases and the equivalent connectionist knowledge bases used in the experiments were constructed from datasets. Also, inference aspects are not discussed in this paper.

2.2 Rule merging

Rule merging aspects have been presented in various approaches, but in different contexts. In the context of rule extraction from neural networks and rule learning from datasets (Hadzic & Dillon 2008), rule merging is used to create more general rules and reduce the total number of produced rules. In the context of fuzzy and neuro-fuzzy systems, merging, along with other operations concerning fuzzy sets and rules, is applied for different reasons especially in data-driven fuzzy models. First, in cases that fuzzy models are extracted from large datasets, extraction is partially performed (i.e. for subsets of data) and then a merging process is applied (Siminski 2011). Second, fuzzy rule merging, simplification and deletion may be applied when there is a need for model simplification and rule base reduction (Siminski 2011; Riid &

Rustern 2011). Third, in evolving fuzzy systems, the incremental manner of learning may yield overlapping, redundant, unimportant and obsolete fuzzy sets and rules (Lughofer 2011). Operations such as merging, simplification and deletion of fuzzy sets and rules handle such issues. With merging, in the above contexts, multiple rules are merged into one. So, merging in the aforementioned contexts does not involve conversion of symbolic rules to another formalism.

Our approach lies in a neuro-symbolic context that provides integrated inference, involves available symbolic rules elicited from experts or produced from datasets and reduces rule base size through conversion to an equivalent and more compact formalism by merging symbolic rules.

3. Neurules

3.1 Syntax and Semantics

Neurules are a kind of integrated rules. The form of a neurule is depicted in Figure 1a. Each condition C_i is assigned a number sf_i, called its *significance factor*. Moreover, each rule itself is assigned a number sf_0, called its bias factor. Internally, each neurule is considered as an adaline unit (Figure1b). The inputs C_i (i=1,...,n) of the unit are the conditions of the rule. The weights of the unit are the significance factors of the neurule and its bias is the bias factor of the neurule. Each input takes one of the following discrete values: [1(true), -1(false), 0(unknown)].

The output D, which represents the conclusion of the rule, is calculated via the standard formulas:

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

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

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. The LMS learning algorithm is used to compute the values of the significance factors as well as the bias factor of a neurule. Examples of neurules are shown in Tables 2 and 4.

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 symbol representing a concept in the domain, 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 symbol (e.g. “male”, “night-pain”) or a number (e.g “5”). <bias-factor> and <significance-factor> are (real) numbers.

Let us consider a finite set of *domain variables* V = $\{V_i\}$, $1 \leq i \leq m$, which represent the concepts of the problem domain involved in making inferences. 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*, that is variables for which the user will be prompted to give a value during inference,
- *intermediate or inferable variables*, that is variables constituting intermediate goals of the inference process,
- *output or goal variables*, that is variables constituting the (final) goals of the inference process.

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 variables in its conclusions. An output neurule is one having an output variable in its conclusion.

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, ($C_i^{R_k} \equiv$ "$V_i^{R_k}$ 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, ($D^{R_k} \equiv$ "$V_d^{R_k}$ 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) = { $V_1^{R_k}$, $V_2^{R_k}$, …, $V_{n_{R_k}}^{R_k}$, $V_d^{R_k}$ }.
- *conds*(R_k): the set of conditions of R_k, i.e.
  conds(R_i) = { $C_i^{R_k}$, $1 \leq i \leq n_{R_k}$ }.

3.2 Neurule-Based System Architecture

In Figure 2, the architecture of a neurule-based system is illustrated. The run-time system (in the dashed rectangle) consists of four modules: the *neurule base (NRB)*, the *hybrid inference engine*, the *working memory* and the *explanation mechanism*. These modules are more or less functionally similar to those of a conventional rule-based system.

The neurule base contains neurules alongside certain information concerning their source knowledge (e.g. symbolic rules). The hybrid inference engine performs inferences by taking into account the data in the working memory and the neurules in neurule base. The working memory contains *fact assertions* either given by the user (as initial input data or during an inference course), or produced by the system, as intermediate or final conclusions during an inference course. The contents of the

neurule base are produced from a *symbolic rule base* by applying the *conversion mechanism*.

4. Merging Symbolic Rules into Neurules

One way of producing a neurule base (NRB) is by conversion from a (propositional type) symbolic rule base (SRB). A symbolic rule consists of a conjunction of conditions and a conclusion. Examples of symbolic rules are shown in Tables 1 and 5 where "," (as already mentioned) denotes conjunction. Existing SRBs (of propositional type) can be easily transformed into an SRB of the above syntax and then converted to an NRB. An SRB may be the result of direct knowledge elicitation from experts or the product of an automated knowledge acquisition method. In this way, existing SRBs can be exploited for the production of neurules.

4.1 Conversion Algorithm

The conversion of an SRB to an NRB is achieved by applying the conversion algorithm presented in (Hatzilygeroudis and Prentzas 2000). Application of the conversion algorithm does not result in a refinement of the converted SRB. It creates an equivalent knowledge base (NRB) whose size is less than that of SRB. The conversion algorithm tries to merge all symbolic rules having the same conclusion into one neurule. However, this is not always possible, due to non-linearity problems, as explained later in this section. In any case, each produced neurule usually is the result of merging two or more symbolic rules.

To be able to construct a neurule base from an existing symbolic rule base, we need two types of data: (a) a set of domain variables, with their possible values and (b) a symbolic rule base.

We introduce the following definitions (symbol '≡' stands for 'identical'):

Definition 1. Two conditions C_i and C_j are *related* if $V_i \equiv V_j$.

Definition 2. A set of symbolic rules MS={R_1, R_2,…, R_m}, m≥1 is called a *merger set* if $D^{R_i} \equiv D^{R_j}$ ∀ i≠j, 1≤i≤m, 1≤j≤m.

Definition 3. A non-merging rule is a symbolic rule with a unique conclusion in the SRB.

Definition 4. The closeness (R-closeness) between two symbolic rules R_m, R_j is defined as follows: $R-closeness(R_m, R_j) = \sum \delta(C_i^{R_m}, C_l^{R_j})$ where $\delta(C_i^{R_m}, C_l^{R_j}) = \begin{cases} 1, C_i^{R_m} \equiv C_l^{R_j} \\ 0, \text{otherwise} \end{cases}$.

The *conversion algorithm* is outlined as follows.

Input:

(a) A set of domain variables V and corresponding value sets S_{V_i}, $V_i \in V$,

(b) Information concerning groups of rule conditions that due to pragmatic reasons cannot be simultaneously true or false,
(c) A symbolic rule base (SRB)

Output:

A set of neurules (NRB)

Body

1. Group symbolic rules into (initial) merger sets.
2. For each merger set,
2.1 Construct a merger
2.2 Produce a training set for the merger
2.3 Train the merger individually
2.4 If training is successful, produce the corresponding neurule.
2.5 Otherwise, split the merger set into two disjoint subsets and execute recursively Steps 2.1-2.5 for each subset.

The initial merger sets contain all rules of the SRB having the same conclusion. A merger is a neurule having as conditions all the conditions of the symbolic rules in the corresponding merger set without duplications and significance factors as well as bias factor set to a proper initial value. For each merger, a training set is extracted from the truth table of the combined logical function of the rules in the set (the disjunction of the conjunctions of the conditions of each rule).

Definition 5. The output of the combined logical function of the symbolic rules in a merger set MS={R_1, R_2,…, R_m}, m≥1 is produced as follows:

$$F = \left(C_1^{R_1} \wedge C_2^{R_1} \wedge \ldots \wedge C_{n_{R_1}}^{R_1}\right) \vee \left(C_1^{R_2} \wedge C_2^{R_2} \wedge \ldots \wedge C_{n_{R_2}}^{R_2}\right) \ldots \vee \left(C_1^{R_m} \wedge C_2^{R_m} \wedge \ldots \wedge C_{n_{R_m}}^{R_m}\right)$$

From the training set, extracted from the truth table of the combined logical function, unacceptable training patterns are eliminated since certain conditions cannot be simultaneously true or false (Hatzilygeroudis and Prentzas 2000).

So, the steps of the conversion algorithm can be analyzed as follows.

1. For each distinct D^{R_k} in SRB

   1.1 Let MS_i={R_i: $D^{R_i} \equiv D^{R_k}$, 1≤i≤m}

2. For each MS_i

   2.1 Conds(MS_i)=*conds*(R_1) ∪ *conds*(R_2) ∪ … ∪ *conds*(R_m)

   2.2 Construct a merger $\mathcal{R}_j$ with

   $$D^{R_j} \equiv D^{R_k}$$

   $$conds(\mathcal{R}_j) \equiv \{\text{“}C_l^{R_j}(0)\text{”}: C_l^{R_j} \in \text{Conds}(MS_i)\}$$

   2.3 Produce an initial training set S_k for merger $\mathcal{R}_j$, consisting of *n* patterns that are $(n_{R_j}+1)$-tuples:

   $$S_k = \{ t_i^k \},\ t_i^k = < v_{i_1}^k,\ v_{i_2}^k,\ \ldots,\ v_{i_{n_{R_j}}}^k,\ v_{i_d}^k >,\ 1 \le i \le n$$

   where each $v_{i_l}^k$ corresponds to $C_l^{R_j}$ and $v_{i_d}^k$ corresponds to D^{R_j}.

   Also, $v_{i_d}^k \leftarrow 1$, if $F\left(v_{i_1}^k, v_{i_2}^k, \ldots, v_{i_{n_{R_j}}}^k\right)$=true, otherwise $v_{i_d}^k \leftarrow -1$.

   2.4 Filter any pattern $t_i^k : \exists\ \left(v_{i_l}^k, v_{i_m}^k\right) = (1,1)$ and $V_{C_l}^{R_j} \equiv V_{C_m}^{R_j}$

2.5 Filter any pattern t_i^k: $\exists \left(v_{i_l}^k, v_{i_m}^k, \ldots, v_{i_r}^k\right) = (-1, -1, \ldots, -1)$ and $V_{C_l}^{R_j} \equiv V_{C_m}^{R_j} \equiv \ldots \equiv V_{C_r}^{R_j}$ and $S_{V_{C_l}^{R_j}} \equiv \left\{v_l^{R_j}\right\} \cup \left\{v_m^{R_j}\right\} \cup \ldots \cup \left\{v_r^{R_j}\right\}$

2.6 Filter any pattern t_i^k that is invalid due to pragmatic reasons

Each merger is individually trained using the standard LMS algorithm. Training of a merger may not be always successful meaning that it cannot always find a set of significance and bias factors that classify correctly all of the training patterns. This happens, if the patterns of the training set are inseparable (as in the case of the patterns corresponding to the XOR function). When training fails, the merger set is split into disjoint subsets producing more than one neurule having the same conclusion. More specifically, splitting a merger (sub)set is guided by a *least closeness pair* (LCP) of symbolic rules (chosen based on a strategy) of the merger set. An LCP of rules in a merger set is a pair of rules that have the minimum R-closeness. We give the definition of an LCP of symbolic rules in the following.

Definition 6. A *least closeness pair* LCP_n of the merger set of symbolic rules $MS=\{R_1, R_2, \ldots, R_m\}$, $m \geq 2$ is a pair of symbolic rules that have the minimum R-closeness:

$$LCP_n = \{(R_i, R_j): i \neq j, R_i, R_j \in MS \wedge \forall k, \forall l, k \neq l, R_k, R_l \in MS \Rightarrow \text{R-closeness}(R_i, R_j) \leq \text{R-closeness}(R_k, R_l)\}$$

So, in case of inseparability, two merger subsets are created each containing as its initial element one of the rules of the LCP, called its pivot. Each of the other rules in the set is distributed between the two subsets based on their closeness to their pivots. That is, each subset contains rules, which are closer to its pivot. If training fails, for a merger of a merger subset, the corresponding subset is further split into two other subsets, based on one of its LCPs. This continues, until training succeeds or the merger subset contains only one rule. This kind of splitting stems from the observation that separable sets have rules with larger average closeness than inseparable ones.

In the following, we give the rest steps of the conversion algorithm involving training and splitting.

2.7 Train $\mathcal{R}_j$ using the LMS algorithm with S_k as the training set.

2.8 If training is successful or $|MS_i|=1$, produce $\mathcal{R}_j$ with the calculated $sf_i^{R_j}$ and stop (success).

2.9 Find an LCP $\equiv (R_{k1}, R_{k2})$.

2.10 Split MS_i into MS_i^1 and MS_i^2, such that $R_{k1} \in MS_i^1$, $R_{k2} \in MS_i^2$ and $\forall R_m \in MS_i^1$, $\forall R_l \in MS_i^2$, R-closeness(R_m, R_{k2}) < R-closeness(R_m, R_{k1}) and R-closeness(R_l, R_{k2}) > R-closeness(R_l, R_{k1}).

2.11 Apply steps 2.1 to 2.11 for $MS_i = MS_i^1$ and $MS_i = MS_i^2$ separately.

4.2 Conversion Example

We present an example of applying the conversion algorithm. We use the merger set shown in Table 1 that consists of four symbolic rules $\{R_1, R_2, R_3, R_4\}$ taken from a medical diagnosis rule base. The domain variables contained in the conditions of those rules take values from the following discrete sets: $S_{patient}$={human0-20,

human21-35, human36-55, human56}, S_{fever}={no-fever, low, medium, high}, S_{pain}={night, continuous}, $S_{ant\text{-}reaction}$={low, medium, high}.

The merger of this merger set (shown in Table 2) contains the nine distinct conditions of the four rules. The training set of the merger is extracted from the truth table of the combined logical function of the rules of the merger set: $F = (C_1 \wedge C_2 \wedge C_3) \vee (C_1 \wedge C_3 \wedge C_4 \wedge C_5) \vee (C_6 \wedge C_7 \wedge C_8) \vee (C_2 \wedge C_3 \wedge C_9)$, where C_1≡patient is human0-20, C_2≡fever is high, C_3≡pain is night, C_4≡fever is no-fever, C_5≡ant-reaction is medium, C_6≡patient is human21-35, C_7≡fever is medium, C_8≡pain is continuous, C_9≡patient is human36-55.

Table 3 depicts certain patterns corresponding to the truth table of the combined logical function after having eliminated invalid patterns.

Patterns with the following characteristics are eliminated as invalid:

- At least two of the columns $\{v_{i_1}^k, v_{i_6}^k, v_{i_9}^k\}$ corresponding to conditions C_1, C_6 and C_9 are simultaneously true.
- At least two of the columns $\{v_{i_2}^k, v_{i_4}^k, v_{i_7}^k\}$ corresponding to conditions C_2, C_4 and C_7 are simultaneously true.
- Columns $\{v_{i_3}^k, v_{i_8}^k\}$ corresponding to conditions C_3 and C_8 are simultaneously true.
- Columns $\{v_{i_3}^k, v_{i_8}^k\}$ corresponding to conditions C_3 and C_8 are simultaneously false (since S_{pain}={night, continuous}).

The training patterns of the training set are inseparable and the initial merger set is split in two subsets: MS_1={R_1, R_2, R_4} and MS_2={R_3}. The LCP that guides splitting is (R_1, R_3). Notice that R-closeness(R_1, R_2) > R-closeness(R_3, R_2) and R-closeness(R_1, R_4) > R-closeness(R_3, R_4).

The merger of MS_1 contains the six distinct conditions of the three rules C_{11}≡patient is human0-20, C_{12}≡fever is high, C_{13}≡pain is night, C_{14}≡fever is no-fever, C_{15}≡ant-reaction is medium, C_{16}≡patient is human36-55. The training set of the merger is extracted from the truth table of the combined logical function of the rules of the merger set: $F = (C_{11} \wedge C_{12} \wedge C_{13}) \vee (C_{11} \wedge C_{13} \wedge C_{14} \wedge C_{15}) \vee (C_{12} \wedge C_{13} \wedge C_{16})$.

Training of the merger of MS_1 is successful and neurule NR_1-R_2-R_4 is produced. Rule R_3 is converted to a neurule (i.e. NR_3). So, from the initial merger set of four symbolic rules, two neurules are produced. Table 4 depicts the produced neurules.

Table 5 depicts in the form of a matrix the corresponding connectionist knowledge base constructed from the training set extracted from the truth table of the combined logical function of the rules of the merger set shown in Table 1. The connectionist knowledge base contains two random cells. According to the instructions given in (Gallant 1993), input nodes are connected to both random cells and the output node. Symbols ‘uA’ and ‘uB’ are used, as in (Gallant 1993), to represent the two random cells. Figure 3 graphically depicts the connectionist network. Corresponding neurules, shown in Table 4, are more natural as they involve only domain concepts and have a propositional form. The construction of neurules required much less effort than the construction of the connectionist network. We tried to construct connectionist networks with a different number of random cells in order to find a structure that would classify all training examples. More specifically, we tried to construct connectionist networks with three to ten random cells but without success. Only a network with two random cells was able to successfully classify all training examples.

5. Alternative Splitting Methods

In the conversion process described in the previous sections, splitting a merger set into two disjoint subsets was guided by LCPs. However, alternative splitting methods may be employed. A splitting method regards implementation of steps 2.9-2.10 of the conversion algorithm. In this section, we present certain alternative splitting methods.

5.1 Splitting Method based on 2-Medoids

The first alternative splitting method is based on the k-medoids algorithm and more specifically on the 2-medoids algorithm. The k-medoids algorithm is a clustering algorithm related to the k-means algorithm. Both algorithms partition a set of objects into k clusters. This is done by attempting to maximize the closeness (or minimize the distance) between objects assigned to a cluster and the center of the specific cluster. The value of parameter k (i.e. the number of clusters) is known a priori. The main difference between the k-means and the k-medoids algorithm is related to the way they select the centers of clusters. The k-means algorithm computes the means of clusters and assigns them as their centers. The k-medoids algorithm selects centers that are actual objects of the set. These centers are called medoids.

According to the most common realization of k-medoids (i.e. the Partitioning Around Medoids algorithm, Theodoridis & Koutroumbas 2008), initially k objects in the set are selected as medoids and corresponding initial clusters are formed by assigning each object in the set to the closest medoid. Each medoid is then swapped with each non-medoid object and the total sum of closeness to medoids for all objects in clusters is computed. Medoids maximizing the total sum of closeness are selected and objects are clustered accordingly. This is repeated until there is no change in the medoids.

We experimented with a variation of the 2-medoids algorithm for two main reasons: (a) the rules in a merger set may contain an unequal number of conditions meaning that they may involve different variables and (b) the 2-medoids algorithm is relevant to the splitting process guided by an LCP in which rules are organized (or clustered) into subsets having as centers actual objects (i.e. rules) of the set.

In our case, we employed the 2-medoids algorithm as the value of k was not known a priori. More specifically, in case a merger set cannot be converted into a single neurule, splitting of the merger set into two subsets is performed according to the 2-medoids algorithm. Only step 2.10 of the conversion process mainly changes. The other steps remain as before with a minor change to step 2.9. More specifically, steps 2.9 and 2.10 of the conversion process become as follows: Find an LCP $p_k \equiv (R_{k1}, R_{k2})$.

2.9 Let rules R_{k1} and R_{k2} be the initial medoids.

2.10 Split MS_i in two subsets as follows:

2.10.1 Split MS_i into MS_i^1 and MS_i^2, such that $R_{k1} \in MS_i^1$, $R_{k2} \in MS_i^2$ and $\forall R_m \in MS_i^1$, $\forall R_l \in MS_i^2$, R-closeness(R_m, R_{k2}) < R-closeness(R_m, R_{k1}) and R-closeness(R_l, R_{k2}) > R-closeness(R_l, R_{k1}).

2.10.2 Compute $close\_sum = \sum_{R_m \in MS_i} \max[R - closeness(R_m, R_{k1}), R - closeness(R_m, R_{k2})]$, $R_m \neq R_{k1}$ and $R_m \neq R_{k2}$

2.10.3 For each $R_{p1} \in MS_i$

Compute $close\_sum = \sum_{R_m \in MS_i} \max\left[R-closeness(R_m, R_{p1}), R-closeness(R_m, R_{k2})\right]$,

$R_{p1} \neq R_{k1}$, $R_{p1} \neq R_{k2}$, $R_m \neq R_{p1}$ and $R_m \neq R_{k2}$

2.10.4 For each $R_{p2} \in MS_i$

Compute $close\_sum = \sum_{R_m \in MS_i} \max\left[R-closeness(R_m, R_{k1}), R-closeness(R_m, R_{p2})\right]$,

$R_{p2} \neq R_{k1}$, $R_{p2} \neq R_{k2}$, $R_m \neq R_{p2}$ and $R_m \neq R_{k1}$

2.10.5 Select the pair (R_{k1}, R_{k2}), (R_{p1}, R_{k2}) or (R_{k1}, R_{p2}), $R_{p1} \in MS_i$, $R_{p2} \in MS_i$ having the maximum close_sum. Let (R_{med1}, R_{med2}) be this pair where $R_{med1} \in MS_i$ and $R_{med2} \in MS_i$.

2.10.6 If $(R_{med1}, R_{med2}) \neq (R_{k1}, R_{k2})$, repeat steps 2.10.1-2.10.6 for $R_{k1} \equiv R_{med1}$ and $R_{k2} \equiv R_{med2}$

The two initial medoids could be the members of the LCP that would have guided splitting, if splitting was performed purely based on the LCP. In section 6, we also experimented with members of maxsum pair (see following section) as initial medoids.

5.2 Splitting Method based on Maxsum Pair

The second alternative splitting method is not a repetitive process as the one based on 2-medoids. More specifically, the rule R_{p1} having the maximum average closeness to the other rules in MS_i is found. Then a rule R_{p2} in MS_i that is not very close to R_{p1} is chosen and splitting is guided by the pair (R_{p1}, R_{p2}) mentioned as 'maxsum pair' in the following. So steps 2.9 - 2.10 of the conversion algorithm become as follows:

2.9 Let $R_{p1} \in MS_i$ be the rule having the maximum average closeness to the other rules in MS_i.

2.10.1 Let $S_{R_{p1}} = \{R_i \in MS_i$: R-closeness($R_i$, R_{p1}) $\leq$ min(|conds(R_i)|, |conds(R_{p1})|) – 3$\}$.

2.10.2 If $S_{R_{p1}} \equiv \emptyset$, let $S_{R_{p1}} = \{R_i \in MS_i$: R-closeness($R_i$, R_{p1}) = min(|conds(R_i)|, |conds(R_{p1})|) – 2$\}$.

2.10.3 Find the rule $R_{p2} \in S_{R_{p1}}$ having the maximum average closeness to the other rules in MS_i.

2.10.4 Split MS_i into MS_i^1 and MS_i^2, such that $R_{p1} \in MS_i^1$, $R_{p2} \in MS_i^2$ and $\forall R_m \in MS_i^1$, $\forall R_l \in MS_i^2$, R-closeness(R_m, R_{p2}) < R-closeness(R_m, R_{p1}) and R-closeness(R_l, R_{p2}) > R-closeness(R_l, R_{p1}).

6. Mergability Aspects

6.1 Mergability Criteria

An aspect of interest in the above process concerns introduction of criteria concerning the 'mergability' of a merger set that is, determining whether a merger set can (or cannot) be converted (or merged) into a single neurule, without using training. By determining in advance whether a merger set cannot be converted into a single neurule, training of the corresponding merger can be omitted. In such cases, splitting could be directly performed, without training the mergers. So, the time required to convert a symbolic rule base into a neurule base would decrease, since certain trainings would be omitted.

In the above example (rules of Table 1), three trainings concerning the mergers of the merger (sub)sets $\{R_1, R_2, R_3, R_4\}$, $\{R_1, R_2, R_4\}$ and $\{R_3\}$ were performed. Two of the trainings, those corresponding to merger subsets $\{R_1, R_2, R_4\}$ and $\{R_3\}$, were successful and resulted in the production of neurules. By avoiding the training corresponding to merger set $\{R_1, R_2, R_3, R_4\}$ and simply splitting the set to subsets $\{R_1, R_2, R_4\}$ and $\{R_3\}$, conversion would have taken less time.

Besides conversion time gains, determining whether a merger set can be converted into a single neurule is of general interest from a representational point of view. More specifically, it would be interesting to determine criteria of whether a set of symbolic rules can (or cannot) be converted into a single, equivalent but more efficient neuro-symbolic rule. This is attempted in the following.

A merger corresponding to a merger set containing a single symbolic rule can be successfully trained, since its training set corresponds to a conjunction, and represents a separable set. So, the interest goes to merger sets containing at least two symbolic rules. It should be mentioned that the rules in a merger set may contain an unequal number of conditions. We define criteria guided from experimental results. The criteria are based on R-closeness of rule pairs in a merger set. In certain criteria, we distinguish between merger sets with rules that contain related conditions and merger sets with rules that do not contain related conditions. Some criteria apply to both types of merger sets. Examples regarding specific merger sets are also given for the defined criteria. The defined criteria involve merger sets with rules containing at least two conditions.

It should be mentioned that in order to be able to merge a set of rules into a single neurule, corresponding rules should have certain common conditions. More specifically, a pair of symbolic rules without any common condition cannot be merged into a single neurule. Therefore, merger sets containing one or more pairs of such rules cannot be converted into a single neurule. So, we introduce the following criterion:

Mergability criterion 1. *A merger set* $MS=\{R_1, R_2,\ldots,R_m\}$, $m\geq 2$, $|conds(R_i)|\geq 2$ $\forall R_i \in MS$, *cannot be converted to a single neurule if* $\exists (R_i, R_k)$, $R_i, R_k \in MS$ *with* $R\text{-closeness}(R_i, R_k)=0$, $1\leq i\leq m$, $1\leq k\leq m$, $i\neq k$.

Criterion 1 is satisfied by the merger set $\{R_1, R_2, R_3, R_4\}$ of the rules in Table 1, given that $R\text{-closeness}(R_1,R_3)=0$, $R\text{-closeness}(R_2,R_3)=0$ and $R\text{-closeness}(R_3,R_4)=0$. Therefore, this merger set cannot be converted into a single neurule.

According to criterion 1, a requirement that a merger set $MS=\{R_1, R_2,\ldots,R_m\}$, $m\geq 2$ should satisfy so that it could be converted into a single neurule is the satisfaction of the condition: $R\text{-closeness}(R_i, R_k)>0$ $\forall$ $i\neq k$, $1\leq i\leq m$, $1\leq k\leq m$. In order to identify specific positive values for R-closeness of rule pairs in a merger set that might have an effect on its mergability, we conducted a number of experiments. We started with merger sets containing only two rules and then investigated merger sets with at least three rules.

We noticed that any merger set $MS=\{R_1, R_2\}$ with only two rules can be converted into a single neurule if $R\text{-closeness}(R_1, R_2)=\min(|conds(R_1)|, |conds(R_2)|) - 1$. We also noticed that any merger set $MS=\{R_1, R_2,\ldots,R_m\}$, $m\geq 2$ can be converted into a single neurule if the merger set of each pair $\{R_i, R_k\}$ can be converted to a single neurule, $\forall$ $i\neq k$, $1\leq i\leq m$, $1\leq k\leq m$. As explained in the following, this condition is less strict for merger sets with rules that have related conditions. However, merger sets whose rules do not have any related conditions can be converted into a single neurule only if the

merger set of each pair of rules can itself be converted into a single neurule. Therefore, we introduce the following criterion:

Mergability criterion 2. *A merger set MS={R_1, R_2,…,R_m}, m≥2, |conds(R_i)|≥2 ∀R_i∈MS,* whose rules do not have any related conditions *can be converted into a single neurule only if R-closeness(R_i, R_k) = min(|conds(R_i)|, |conds(R_k)|) – 1, ∀ i≠k, 1≤i≤m, 1≤k≤m.*

For instance, this second criterion is satisfied by the merger set {R_1, R_2, R_3, R_4} of rules in Table 6. So, the merger set can be converted into a single neurule (shown in Table 7). Notice that those four symbolic rules do not have related conditions.

As mentioned, a merger set MS={R_1, R_2,…,R_m}, m≥2 whose rules have related conditions can be converted into a single neurule even if R-closeness(R_i,R_k) < min(|conds(R_i)|, |conds(R_k)|) – 1 for some rules R_i, R_k, i≠k, 1≤i≤m, 1≤k≤m. This happens because certain training patterns are excluded from the merger's training set as invalid.

So, criterion 2 for mergers with rules having related conditions becomes as follows:

Mergability criterion 2A. *A merger set MS={R_1, R_2,…,R_m}, m≥2, |conds(R_i)|≥2 ∀R_i∈MS, containing rules having related conditions can be converted into a single neurule if R-closeness(R_i,R_k) = min(|conds(R_i)|, |conds(R_k)|) – 1, ∀ i≠k, 1≤i≤m, 1≤k≤m.*

Furthermore, experiments showed that a merger set containing rules with related conditions, with most pairs of rules (R_m, R_j) having R-closeness(R_m, R_j)=min(|conds(R_m)|, |conds(R_j)|) – 1 and with certain pairs of rules (R_i, R_k) having R-closeness(R_i, R_k)=min(|conds(R_i)|, |conds(R_k)|) – 2 can be converted into a single neurule. So, we introduce the following two criteria.

Mergability criterion 3. *A merger set MS={R_1, R_2,…,R_m}, m≥2, |conds(R_i)|≥3 ∀R_i∈MS, containing rules having related conditions cannot be converted to a single neurule if the following is satisfied:*
∃(R_i,R_k), R_i,R_k∈MS with R-closeness(R_i,R_k)< min(|conds(R_i)|, |conds(R_k)|) – 2, 1≤i≤m, 1≤k≤m, i≠k.

Mergability criterion 4. *A merger set MS={R_1,R_2,…,R_m}, m≥2, |conds(R_i)|≥3 ∀R_i∈MS, containing rules having related conditions, cannot be converted into a single neurule if the following are satisfied:*
i) ∀R_i,R_k∈MS, R-closeness(R_i, R_k) ≥ min(|conds(R_i)|, |conds(R_k)|) – 2, 1≤i≤m, 1≤k≤m, i≠k and
ii) $|S_{P1}|<|S_{P2}|$, where S_{P1}={(R_i, R_k): R_i, R_k∈MS, R-closeness(R_i, R_k)=min(|conds(R_i)|, |conds(R_k)|)-1, 1≤i≤m, 1≤k≤m, i≠k} and S_{P2}={(R_i, R_k): R_i, R_k∈MS, R-closeness=min(|conds(R_i)|, |conds(R_k)|) – 2, 1≤i≤m, 1≤k≤m, i≠k}.

It should be mentioned that in case of merger sets with rules having related conditions, the first criterion is subsumed by the third criterion. In the following, we give examples for the third and fourth criterion.

The third criterion is satisfied by the merger set {R_1, R_2, R_3, R_4} of rules in Table 1. So, this merger set cannot be converted into a single neurule.

Furthermore, by the merger subset {R_1, R_2, R_4} neither third nor fourth criterion is satisfied. More specifically, condition (i) of the fourth criterion is satisfied. However,

S_{P1} = {(R_1, R_2), (R_1, R_4)} and S_{P2} = {(R_2, R_4)}, so $|S_{P1}| > |S_{P2}|$, since $|S_{P1}|=2$ and $|S_{P2}|=1$. Thus, mergability criteria cannot give indications that the merger set cannot be converted into a single neurule. Indeed, after training, neurule NR_1-R_2-R_4 (shown in Table 2) is produced.

6.2 Conversion Process Benefits

By checking the satisfaction of the mergability criteria, the conversion algorithm is improved, given that certain (unnecessary) training effort may be omitted. This is based on the indications about merger sets provided by the mergability criteria, which are the following:

(a) A merger set can be converted into a single neurule (satisfaction of the second criterion).

(b) A merger set cannot be converted into a single neurule (satisfaction of the third or fourth criterion for merger sets which contain rules having related conditions and all rules have at least three conditions and; second criterion is not satisfied for merger sets which contain rules not having related conditions; satisfaction of first criterion for merger sets containing certain rules with two conditions).

(c) There is no indication that a merger set which contains rules having related conditions cannot be converted into a single neurule (first, third and fourth criterion are not satisfied).

So, the conversion process has been revised to take into account the indications provided by the mergability criteria. This is done by embedding the following steps:

1. If the second criterion is satisfied by the merger set, then the merger set can be converted to a single neurule.
2. Else if the rules in the merger set have no related conditions, then the merger set cannot be converted to a single neurule.
3. Else if the rules in the merger set have at least two conditions and the first criterion is satisfied, then the merger set cannot be converted to a single neurule.
4. Else if the rules in the merger set have related conditions and all rules have at least three conditions and the third criterion is satisfied, then the merger set cannot be converted to a single neurule.
5. Else if the rules in the merger set have related conditions and all rules have at least three conditions and the fourth criterion is satisfied, then the merger set cannot be converted to a single neurule.
6. Else if none of the above is satisfied, then there is no indication from the mergability criteria that the specific merger set cannot be converted to a single neurule.

7. Experimental Results

In this section, experimental results are presented. They refer to the following:

- Neurule bases are compared with corresponding connectionist knowledge bases.
- The splitting method based on LCPs is compared with the two alternative approaches: the splitting method based on 2-medoids and the splitting method based on maxsum pair.
- The improvement to the conversion process from applying the introduced mergability criteria.

The conversion process involving splitting methods and mergability criteria was implemented in C using MS Visual Studio (Express Edition). The experiments were

run on a computer having a Core(TM)2 Duo CPU, 4 GB RAM and MS Windows 7 installed.

Prior to the presentation of the experimental results, we briefly discuss the characteristics of the rule bases used in the experiments and involved problem domains.

7.1 Rule Base Characteristics used in Experiments

The conversion process was applied to five symbolic rule bases. Three of the symbolic rule bases were elicited from domain experts and the other two were artificially constructed. Two of the rule bases elicited from experts, denoted hereafter by RB1 and RB2, concern medical diagnosis. The third one, denoted by RB3, concerns a banking domain (i.e. credit scoring). The rules in the ensuing merger sets had related conditions. We briefly outline aspects related to the medical and banking rule bases-domains. We also outline aspects related to the two artificial rule bases.

RB1 and RB2 contain sixty-eight (68) and one-hundred-thirty (130) symbolic rules respectively. They concern bone disease diagnosis (Hatzilygeroudis et al. 1997). Rules in RB1 involve fourteen input variables (i.e. gender, age, pain, fever, antinflam-reaction, joints-pain, injury-history, lesion-regions, PSA, PSA-value, menopause, bone-density-measure, bone-density-increase and density-elevated-values), an intermediate variable (i.e. patient-class) and an output variable (i.e. disease). The intermediate variable takes twelve (12) discrete values, whereas the output variable takes sixteen (16) discrete values. Rules in RB2 involve five input variables (i.e. arterial-concentration, capillary-concentration, venous-concentration, blood-concentration, scan-concentration) and an output variable (i.e. disease). 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 banking domain concerns a type of credit scoring. Credit scoring is an important task in times of financial crisis and is a process rating the creditworthiness of persons, corporations, banks, financial institutions and countries (Chuang & Huang 2011; Sreekantha & Kulkarni 2012). An important aspect in credit scoring concerns evaluation of loan applications. Lenders use credit scoring methods to evaluate loan applications and discriminate between risky ones that may result to financial losses and promising ones that are likely to bring in revenue. The specific rule base we experimented with concerns 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 (i.e. net annual income, overall financial status, number of depending children, age, social status), two intermediate variables (i.e. applicant and warrantor assessment) and an output variable (i.e. overall applicant assessment). All variables take discrete values. We used one hundred and seventy three (173) of the constructed rules in our experiments and in the conversion process we assumed that values are non-fuzzy.

The fourth rule base, denoted by RB4, consists of 651 symbolic rules. RB4 involves seven variables, six input and one output. The output variable takes four discrete values. Four of the input variables take four discrete values and two input variables take three discrete values.

The fifth rule base, denoted by RB5, consists of 3322 symbolic rules. RB5 involves nine variables, eight input and one output. The input variables take from two to five

discrete values. More specifically, one input variable takes two discrete values, four input variables take three discrete values, two input variables take four discrete values and one input variable takes five values. The output variable takes four discrete values.

Table 8 summarizes characteristics of the symbolic rule bases (SRBs). Non-merging symbolic rules are the ones having a unique conclusion and thus cannot be merged with any other symbolic rule. The number of initial merger sets corresponds to the number of different conclusions in the rule base.

7.2 Experimental Results Concerning Connectionist Expert Systems

Experiments were conducted involving connectionist expert systems. More specifically, we constructed three connectionist knowledge bases according to the guidelines described in (Gallant 1993) and using the dependency information provided by the rules in RB1, RB2 and RB3 and the training sets used to train the initial merger sets. Compared to equivalent neurule bases, disadvantages mentioned in previous sections of this paper and in previous work of the authors also apply to these three connectionist knowledge bases. For example, much more effort was required to construct the three connectionist knowledge bases compared to the three equivalent neurule bases. At least three working days were required to construct the connectionist knowledge bases, as the overall construction process is neither straightforward nor automated. The construction of each connectionist knowledge base, involved construction of partial networks corresponding to the initial merger sets. Obviously extra work was required for initial merger sets with inseparable training examples. The combined partial networks formed each connectionist knowledge base. Table 9 depicts aspects involving the constructed connectionist knowledge bases. The terms ‘CKB-RB1’, ‘CKB-RB2’ and ‘CKB-RB3’ in the table stand for connectionist knowledge bases constructed from RB1, RB2, RB3, respectively. The table depicts the total number of intermediate and output nodes of the connectionist knowledge bases. In parentheses, the number of random cells of the connectionist knowledge bases is shown. The table also depicts for each connectionist knowledge base, the number of partial networks that had to be constructed and combined to form the connectionist knowledge base. In parentheses, the number of initial merger sets with inseparable training examples is also depicted.

The computational cost of the neurule-based inference mechanism and the inference mechanism used in connectionist expert systems has been compared in (Hatzilygeroudis & Prentzas 2010). As shown in (Hatzilygeroudis & Prentzas 2010), the inference mechanism in (Ghalwash 1998) performs better than the inference mechanism in (Gallant 1993) in terms of computational cost. The computational time required by the inference mechanism in (Ghalwash 1998) for CKB-RB1, CKB-RB2 and CKB-RB3 is roughly 1.50, 2.50 and 2.64 times greater than the computational time required by the neurule-based inference mechanism for the corresponding neurule bases.

7.3 Experimental Results Concerning the Alternative Splitting Methods

Experiments were also conducted for comparing the different approaches to splitting in the conversion process. Comparison is made in terms of the number of produced neurules. Two variations of the conversion process based on 2-medoids splitting were implemented. The one used LCP members as initial medoids, whereas the other used members of the maxsum pair. Table 10 presents the corresponding results for the five rule bases, RB1, RB2, RB3, RB4 and RB5.

Results demonstrate that the splitting method based on the members of the maxsum pair generally performs worse than the other methods. Furthermore, the 2-medoids splitting performs better when the medoids are initialized to LCP members instead of members of maxsum pairs. The 2-medoids splitting with the medoids initialized to LCP members seems to perform better than pure LCP splitting in most cases. Pure LCP splitting may perform better than the 2-medoids splitting with the medoids initialized to members of maxsum pairs in certain cases. So, it seems worth trying pure LCP splitting and the 2-medoids splitting method with medoids initialized to LCP members when converting a symbolic rule base to a neurule base.

7.4 Experimental Results Concerning the Mergability Criteria

Experiments were finally conducted to demonstrate the improvement in the conversion process by applying the mergability criteria. More specifically, the criteria were applied to indicate whether a merger set cannot be converted to a single neurule and omit training effort. The conversion process was applied to the five symbolic rule bases. The rules in the ensuing merger sets had related conditions. We applied the first, third and fourth criteria (in this order) to save useless trainings. We could have avoided applying the first criterion, given that it is subsumed by the third one, but we wanted to record the percentage of merger sets satisfying it.

Tables 11-14 summarize characteristics of the rule bases and the conversion process for each of the alternative splitting methods. More specifically, Table 11 provides results for RB1 and RB2, Table 12 provides results for RB3, Table 13 provides results for RB4 and Table 14 provides results for RB5. The tables depict separately the number of non-merging symbolic rules and the number of merger sets that (after splitting) have a single rule. The number of produced neurules is the sum of the number of non-merging rules, the number of merger sets having a single rule and the number of total merger trainings subtracting the number of merger trainings that would have failed (not producing neurules). Entry “8 (4/3/1)” in the last row means on the one hand that in total the criteria indicate that eight trainings would have failed and on the other hand, four, three and one of those trainings were indicated by the first, third and fourth criterion respectively. By applying the defined criteria, the merger trainings that would have failed are indicated and thus omitted. By applying the criteria in the conversion of RB1, RB2, RB3, RB4 and RB5 roughly 27-33%, 55-57%, 50-52%, 50% and 59-64% of trainings respectively are omitted (not taking into account merger sets having a single rule). All of the three criteria are useful in the conversions of RB1, RB2 and RB3. In the conversions of RB4 and RB5**,** the first criterion was not applicable as in every initial merger set each rule had at least one common condition with any other rule.

8. Conclusion

In this paper, we discuss issues concerning the conversion (i.e. merging) process of symbolic rules to neurules, a type of integrated neuro-symbolic rules. The conversion process tries to merge each set of symbolic rules having the same conclusion into a single neurule. Whenever this is not possible, splitting of the initial merger set of rules is performed according to specific strategies and the ensuing merger subsets of symbolic rules are merged into neurules.

First, we present a comparison between constructing neurule bases and corresponding connectionist bases. The construction process for connectionist knowledge bases required much more effort compared to the construction process for

neurule bases. The neurule bases are also more natural and modular compared to the equivalent connectionist knowledge bases. Furthermore, as shown in earlier works, the inference process is more efficient in neurule-based expert systems compared to the inference process used in connectionist expert systems.

Second, we present alternative splitting methods, based on clustering approaches. Instead of splitting an inseparable merger rule set based on LCPs, we do it using some type of clustering. Experiments performed on five rule bases show that the splitting method based on the 2-medoids clustering increases merging efficiency in certain cases.

Finally, we define mergability criteria, which determine whether a set of symbolic rules having the same conclusion can (or cannot) be merged into a single neurule. The conversion process of symbolic rules to neurules involves certain trainings not directly resulting in the production of neurules. A practical aspect of the criteria is to identify such trainings, which are then omitted, thus reducing conversion time. The results of the experiments on the five rule bases demonstrated that the defined mergability criteria are effective in omitting trainings not directly resulting in the production of neurules, thus improving the conversion efficiency.

Our future research work concerns improvement of the alternative neurule construction process. More specifically, neurules may be also constructed from available empirical data in the form of training examples and dependency information. This construction process creates training sets and splits them into subsets in case of inseparability. So, a future direction of our work could involve implementation and testing of training subset splitting methods based on clustering algorithms in constructing neurules from empirical data.

References

ABRAHAM, A., A. ZOMAYA, V. WADHAI, R. YAGER, A.K. MUDA, and M. KOEPPEN (eds) (2012) *Proceedings of 12th International Conference on Hybrid Intelligent Systems*, New York: IEEE Press.

BELCIUG, S. and F. GORUNESCU (2013), A hybrid neural network/genetic algorithm applied to breast cancer detection and recurrence, *Expert Systems*, **30**, 243–254.

CASTILLO, O., P. MELIN, and J. KACPRZYK (eds) (2012) Recent advances on hybrid intelligent systems, Studies in Computational Intelligence, vol. **451**, Berlin/Heidelberg, Germany: Springer-Verlag.

CHATTOPADHYAY, S. Neurofuzzy models to automate the grading of old-age depression, *Expert Systems* (in press), DOI: 10.1111/exsy.12000.

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

GALLANT, S.I. (1993) *Neural Network Learning and Expert Systems*, Boston, MA: MIT Press.

GARCEZ d'AVILA, A.S., K. BRODA, and D.M. GABBAY (2002) *Neural-symbolic learning systems: Foundations and Applications*, Perspectives in Neural Computing, Berlin/Heidelberg, Germany: Springer-Verlag.

GARCEZ d'AVILA, A.S., L.C. LAMB and D.M. GABBAY (2009) *Neural-Symbolic Cognitive Reasoning*, Berlin/Heidelberg, Germany: Springer-Verlag.

GARCEZ d'AVILA, A.S. and L.C. LAMB (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, Berlin/Heidelberg, Germany: Springer-Verlag, 573–600.

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

HADZIC, F. and T.S. DILLON (2008) Using competitive learning between symbolic rules as a knowledge learning method, in Artificial Intelligence and Practice II, IFIP International Federation for Information Processing, vol. 276, Berlin/Heidelberg, Germany: Springer-Verlag.

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

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

HATZILYGEROUDIS, I. and J. PRENTZAS (2004) Neuro-symbolic approaches for knowledge representation in expert systems, *International Journal on Hybrid Intelligent Systems*, **1**, 111–126.

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

HATZILYGEROUDIS, I. and J. PRENTZAS (eds) (2011a) Combinations of Intelligent Methods and Applications, Smart Innovation, Systems and Technologies, vol. 8, Berlin/Heidelberg, Germany: Springer-Verlag.

HATZILYGEROUDIS, I. and J. PRENTZAS, (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, Berlin/Heidelberg, Germany: Springer-Verlag, 82–91.

HATZILYGEROUDIS, I., P.J. VASSILAKOS and A. TSAKALIDIS (1997) XBONE: a hybrid expert system for supporting diagnosis of bone diseases, in C. Pappas, N. Maglaveras and J.-R. Scherrer (eds) *Proceedings of Medical Informatics Europe*, IOS Press, 295–299.

HOLLDOBLER, S. and Y. KALINKE (1994) Towards a massively parallel computational model for logic programming, in *Workshop on Combining Symbolic and Connectionist Processing*, 68–77; ECAI'94, Amsterdam, The Netherlands.

LIN, X., SUN, J., PALADE, V., FANG, W., WU, X. and XU, W. (2012) Training ANFIS parameters with a quantum-behaved particle swarm optimization algorithm, in *Lecture Notes in Computer Science*, Berlin/Heidelberg, Germany: Springer-Verlag, Vol. 7331, 148–155.

LUGHOFER, E. (2011) Evolving fuzzy systems – Methodologies, advanced concepts and applications, Studies in Fuzziness, vol. 266, Berlin/Heidelberg, Germany: Springer-Verlag, 261–291.

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

PRENTZAS, J. and I. HATZILYGEROUDIS (2011) Neurules – A type of neuro-symbolic rules: an overview, in I. Hatzilygeroudis and J. Prentzas (eds), Combinations of Intelligent Methods and Applications, Smart Innovation, Systems and Technologies, Vol. 8, Berlin/Heidelberg, Germany: Springer-Verlag, 145–165.

PRENTZAS, J. and I. HATZILYGEROUDIS (2012a) Efficiently merging symbolic rules into integrated rules, in *8th International Workshop on Neural-Symbolic Learning and Reasoning*, AAAI-12, AAAI Technical Report WS-12-11, Menlo Park, CA: AAAI Press, 27–32.

PRENTZAS, J. and I. HATZILYGEROUDIS (2012b) An explanation mechanism for integrated rules, in *3rd International Workshop on Combinations of Intelligent Methods and Applications*, I. Hatzilygeroudis and V. Palade (eds), 37–42; ECAI-12, Montpellier, France.

RIID, A. and E. RUSTERN (2011) An integrated approach for the identification of compact, interpretable and accurate fuzzy rule-based classifiers from data, in *Proceedings of 15th International Conference on Intelligent Engineering Systems*, New York: IEEE Press, 101–107.

SAHIN, S. M.R. TOLUN and R. HASSANPOUR (2012) Hybrid expert systems: a survey of current approaches and applications, *Expert Systems with Applications*, **39**, 4609–4617.

SIMINSKI, K. (2011) Merging of fuzzy models for neuro-fuzzy systems, *Theoretical and Applied Informatics*, **23**, 107–126.

D.K. SREEKANTHA, R.V. KULKARNI (2012) Expert system design for credit risk evaluation using neuro-fuzzy logic, *Expert Systems*, **29**, 56–69.

THEODORIDIS, S. and K. KOUTROUMBAS (2008) *Pattern Recognition, 4th Edition*, Burlington, MA: Academic Press.

TOWELL, G. and J. SHAVLIK (1994) Knowledge-based artificial neural networks, *Artificial Intelligence*, **70**, 119–165.

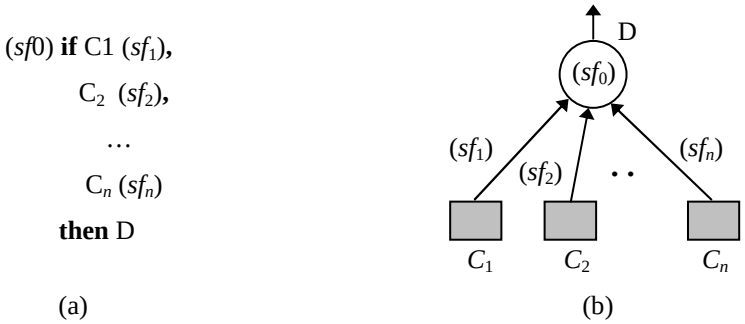

Figure 1. (a) Form of a neurule (b) corresponding adaline unit

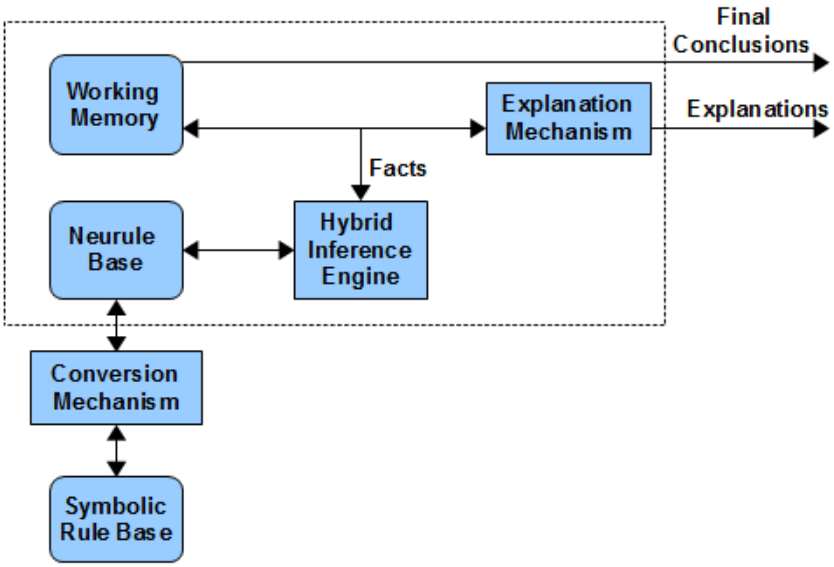

Figure 2. The architecture of a neurule-based system

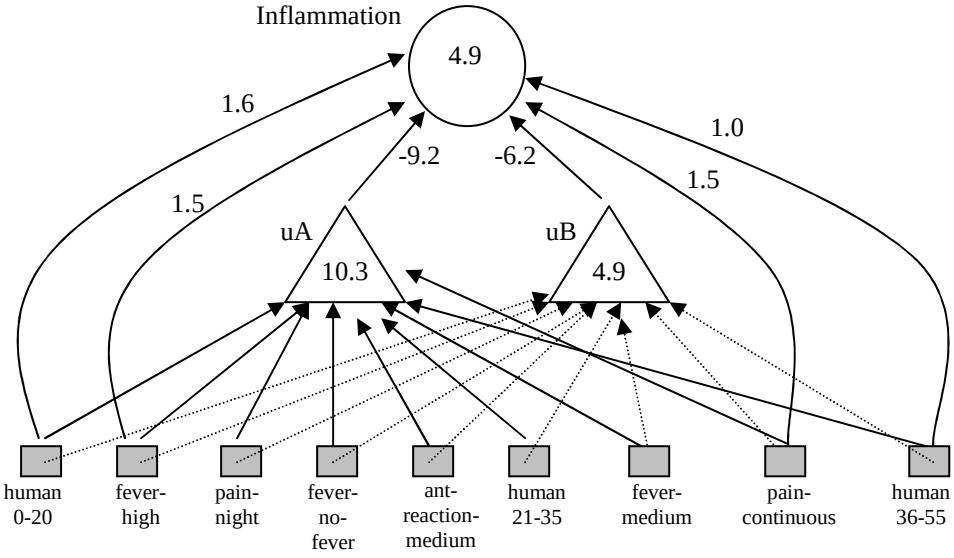

Figure 3. Graphical depiction of the example connectionist knowledge base

Table 1. A set of symbolic rules

R $_1$	***R*** $_3$
if patient is human0-20,	**if** patient is human21-35,
fever is high,	fever is medium,
pain is night	pain is continuous
then disease is inflammation	then disease is inflammation
R $_2$	***R*** $_4$
if patient is human0-20,	**if** patient is human36-55,
fever is no-fever,	fever is high,
ant-reaction is medium,	pain is night
pain is night	**then** disease is inflammation
then disease is inflammation	

Table 2. The merger for merger set {R_1, R_2, R_3, R_4} in Table 1

MR₁-R₂-R₃-R₄
(0) **if** patient is human0-20 (0),
fever is high (0),
pain is night (0),
fever is no-fever (0),
ant-reaction is medium (0),
pain is continuous (0),
fever is medium (0),
patient is human21-35 (0),
patient is human36-55 (0),
then disease is inflammation

Table 3. Certain training patterns used to train the merger of merger set {R_1, R_2, R_3, R_4} in Table 1

$v_{i_1}^k$	$v_{i_2}^k$	$v_{i_3}^k$	$v_{i_4}^k$	$v_{i_5}^k$	$v_{i_6}^k$	$v_{i_{17}}^k$	$v_{i_8}^k$	$v_{i_9}^k$	$v_{i_d}^k$
-1	1	1	-1	-1	-1	-1	-1	-1	-1
-1	1	1	-1	-1	-1	-1	-1	1	1
-1	1	1	-1	-1	1	-1	-1	-1	-1
-1	1	1	-1	1	-1	-1	-1	1	1
-1	1	1	-1	1	1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	1	-1	-1
1	-1	-1	-1	-1	-1	1	-1	-1	-1
1	-1	-1	-1	-1	-1	1	1	-1	-1
1	-1	-1	-1	1	-1	1	1	-1	-1

Table 4. Neurules produced from the merger set in Table 1

***NR*$_1$-*R*$_2$-*R*$_4$**

(-5.6) **if** fever is high (8.7),
pain is night (8.6),
patient is human0-20 (8.2),
patient is human36-55 (5.1),
fever is no-fever (1.5),
ant-reaction is medium (1.3)
then disease is inflammation

***NR*$_3$**

(-2.0) **if** pain is continuous (1.1),
fever is medium (0.8),
patient is human21-35 (0.8)
then disease is inflammation

Table 5. The connectionist knowledge base for the training set extracted from the truth table of the combined logical function of each initial merger

Inflammation											
4.9	1.6	1.5	1.2	-2.0	0.9	1.1	4.4	1.5	1.0	-9.2	-6.2
uA											
10.3	-18.2	-18.3	-15.0	-7.4	-4.5	-0.7	-1.0	-0.3	-11.6	0	0
uB											
4.9	-2.0	-2.1	-2.4	-2.0	0.9	-6.1	-6.4	-5.7	-2.6	0	0
Bias	*human 0-20*	*fever high*	*pain night*	*fever no-fever*	*ant-reaction medium*	*human 21-35*	*fever medium*	*pain continuous*	*human 36-55*	*uA*	*uB*

Table 6. A set of rules that do not contain related conditions

R_1	R_3
if var1 is A1,	**if** var1 is A1,
var3 is C1,	var3 is C1,
var5 is F3,	var5 is F3,
var7 is H2	var7 is H2,
var6 is G1	var8 is I1
var9 is J1	**then** output is D
then output is D	
R_2	R_4
if var1 is A1,	**if** var1 is A1,
var3 is C1,	var2 is B1,
var4 is E3,	var3 is C1
var7 is H2	**then** output is D
then output is D	

Table 7. Neurule produced from the merger set in Table 3

NR_1-R_2-R_3-R_4
(-92.3) **if** var3 is C1 (58.8),
var1 is A1 (55.6)
var2 is B1 (37.5)
var7 is H2 (22.5)
var4 is E3 (12.4),
var5 is F3 (8.3),
var8 is I1 (4.4),
var6 is G1 (1.5),
var9 if J1 (1.0)
then output is D

Table 8. Rule base characteristics

Rule Base and Conversion Characteristics	*RB1*	*RB2*	*RB3*	*RB4*	*RB5*
Number of symbolic rules in SRB	68	130	173	651	3322
Number of non-merging symbolic rules	18	37	0	0	0
Number of initial merger sets (includes number of non-merging symbolic rules)	28	59	11	4	4
Number of input variables	14	5	5	6	8
Number of intermediate variables	1	0	2	0	0
Number of output variables	1	1	1	1	1
Number of distinct conditions	42	21	21	22	27
Number of distinct intermediate conclusions	12	0	6	0	0
Number of distinct final conclusions	16	59	5	4	4

Table 9. Number of nodes contained by connectionist knowledge bases constructed from rule bases RB1, RB2, RB3

Connectionist knowledge base	*Number of nodes*	*Number of partial networks*
CKB-RB1	45 (13)	32 (5)
CKB-RB2	103 (44)	59 (13)
CKB-RB3	89 (78)	13 (11)

Table 10. Comparison of the alternative splitting methods

Splitting Method of the Conversion Process	**RB1**	**RB2**	**RB3**	**RB4**	**RB5**
Conversion Process with pure LCP splitting	***38***	86	72	292	1467
Conversion Process with 2-medoids splitting (medoids initialized to LCP members)	40	***84***	***66***	***258***	***1259***
Conversion Process with 2-medoids splitting (medoids initialized to members of maxsum pairs)	41	89	69	284	1312
Conversion Process with maxsum pair splitting	41	89	72	306	1496

Table 11. Rule base and conversion characteristics for medical rule bases RB1 and RB2

	Pure LCP splitting		***2-medoids splitting (medoids initialized to LCP members)***		***2-medoids splitting (medoids initialized to members of maxsum pairs)***		***Maxsum pair splitting***	
Rule Base and Conversion Characteristics	***RB1***	***RB2***	***RB1***	***RB2***	***RB1***	***RB2***	***RB1***	***RB2***
Number of symbolic rules in SRB	68	130	68	130	68	130	68	130
Number of neurules in equivalent NRB	38	86	40	84	41	89	41	89
Number of non-merging symbolic rules	18	37	18	37	18	37	18	37
Number of initial merger sets (includes number of non-merging symbolic rules)	28	59	28	59	28	59	28	59
Number of total merger trainings (excluding sets with a single rule)	22	51	26	47	27	54	27	54
Number of merger sets having a single rule (excluding non-merging rules)	4	26	4	26	5	29	5	29
Number of merger trainings that would have failed (included in total merger trainings)	6	28	8	26	9	31	9	31
Number of merger trainings that would have failed (as indicated by criteria)	6 (3/3/0)	28 (1/21/6)	8 (4/3/1)	26 (1/19/6)	9 (5/3/1)	31 (1/23/7)	9 (5/3/1)	31 (1/23/7)
Percentage of merger trainings that would have failed (as indicated by criteria)	6/22 = 27%	28/51 = 55%	8/26 = 31%	26/47 = 55%	9/27 = 33%	31/54 = 57%	9/27 = 33%	31/54 = 57%

Table 12. Rule base and conversion characteristics for rule base RB3

	Pure LCP splitting	***2-medoids splitting (medoids initialized to LCP members)***	***2-medoids splitting (medoids initialized to members of maxsum pairs)***	***Maxsum pair splitting***
Rule Base and Conversion Characteristics	***RB3***	***RB3***	***RB3***	***RB3***
Number of symbolic rules in SRB	173	173	173	173
Number of neurules in equivalent NRB	72	66	69	72
Number of non-merging symbolic rules	0	0	0	0
Number of initial merger sets (includes number of non-merging symbolic rules)	11	11	11	11
Number of total merger trainings (excluding sets with a single rule)	118	106	116	121
Number of merger sets having a single rule (excluding non-merging rules)	15	15	11	12
Number of merger trainings that would have failed (included in total merger trainings)	61	55	58	61
Number of merger trainings that would have failed (as indicated by criteria)	61 (7/46/8)	55 (7/44/4)	58 (12/43/3)	61 (14/41/6)
Percentage of merger trainings that would have failed (as indicated by criteria)	61/118=52%	55/106=52%	58/116=50%	61/121=50%

Table 13. Rule base and conversion characteristics for rule base RB4

	Pure LCP splitting	***2-medoids splitting (medoids initialized to LCP members)***	***2-medoids splitting (medoids initialized to members of maxsum pairs)***	***Maxsum pair splitting***
Rule Base and Conversion Characteristics	***RB4***	***RB4***	***RB4***	***RB4***
Number of symbolic rules in SRB	651	651	651	651
Number of neurules in equivalent NRB	291	257	283	305
Number of non-merging symbolic rules	0	0	0	0
Number of initial merger sets (includes number of non-merging symbolic rules)	4	4	4	4
Number of total merger trainings (excluding sets with a single rule)	578	510	562	606
Number of merger sets having a single rule (excluding non-merging rules)	123	67	105	116
Number of merger trainings that would have failed (included in total merger trainings)	287	253	279	301
Number of merger trainings that would have failed (as indicated by criteria)	287 (0/214/73)	253 (0/214/39)	279 (0/231/48)	301 (0/251/50)
Percentage of merger trainings that would have failed (as indicated by criteria)	287/578=50%	253/510=50%	279/562=50%	301/606=50%

Table 14. Rule base and conversion characteristics for rule base RB5

	Pure LCP splitting	***2-medoids splitting (medoids initialized to LCP members)***	***2-medoids splitting (medoids initialized to members of maxsum pairs)***	***Maxsum pair splitting***
Rule Base and Conversion Characteristics	***RB5***	***RB5***	***RB5***	***RB5***
Number of symbolic rules in SRB	3322	3322	3322	3322
Number of neurules in equivalent NRB	1467	1259	1312	1496
Number of non-merging symbolic rules	0	0	0	0
Number of initial merger sets (includes number of non-merging symbolic rules)	4	4	4	4
Number of total merger trainings (excluding sets with a single rule)	2294	2112	2186	2454
Number of merger sets having a single rule (excluding non-merging rules)	636	402	434	534
Number of merger trainings that would have failed (included in total merger trainings)	1463	1255	1308	1492
Number of merger trainings that would have failed (as indicated by criteria)	1463 (0/1092/371)	1255 (0/1014/241)	1308 (0/1069/239)	1492 (0/1262/230)
Percentage of merger trainings that would have failed (as indicated by criteria)	1463/2294=64%	1255/2112=59%	1308/2186=60%	1492/2454=61%