Chapter 18
Rough Neurons: Petri Net Models and Applications

James F. Peters,[1] Sheela Ramanna,[1] Zbigniew Suraj,[2] Maciej Borkowski[1]

[1] University of Manitoba, Winnipeg, Manitoba R3T 5V6, Canada
{jfpeters, sramanna, maciey}@ee.umanitoba.ca
[2] University of Information Technology and Management, Rzeszów,
and University of Rzeszów, Poland
zsuraj@wenus.wsiz.rzeszow.pl

Summary. This chapter presents Petri net models for two forms of rough neural computing: training set production and approximate reasoning schemes (AR schemes) defined in the context of parameterized approximation spaces. The focus of the first form of rough neural computing is inductive learning and the production of training (optimal feature set selection), using knowledge reduction algorithms. This first form of neural computing can be important in designing neural networks defined in the context of parameterized approximation spaces. A high-level Petri net model of a neural network classifier with an optimal feature selection procedure in its front end is given. This model is followed by the development of a number of Petri net models of what are known as elementary approximation neurons (EA neurons). The design of an EA neuron includes an uncertainty function that constructs a granule approximation and a rough inclusion (threshold activation) function that measures the degree to which granule approximation is part of a target granule. The output of an EA neuron is an elementary granule. There are many forms of elementary granules (e.g., conjunction of descriptors, rough inclusion function value). Each of the EA neurons considered in this chapter output a rough inclusion function value. An EA neuron can be designed so that it is trainable, that is, a feedback loop can be included in the design of the EA neuron so that one or more approximation space parameters can be tuned to improve the performance of the neuron. The design of three sample EA neurons is given. One of these neurons behaves like a high-pass filter.

1 Introduction

This chapter introduces a model of rough neurons in the context of rough set theory [16,17] and approximate reasoning schemes (AR schemes) [29]. Studies of neural networks in the context of rough sets [3,6,10,11,14,15,19,22,23,28,29,32,34,35,40], [44] and granular computing [26,29,32,33,35] are extensive. An intuitive formulation of information granulation was introduced by Zadeh [41,42]. Practical applications of rough-neurocomputing have recently been found in predicting urban highway traffic volume [11], speech analysis [14,15], classifying the waveforms of power system faults [6], signal analysis [20], and in assessing software quality [19]. In its most general form, rough-neurocomputing provides a basis for granular computing. A rough mereological approach to a rough neural network springs

from an interest in knowledge synthesized (induced) from successive granule approximations performed by neurons (cooperating agents) [27,28,43]. Such neurons resemble communicating agents described in [12,13]. The distributed agent model for a neural network leads naturally to nonlayered, neural network architectures, and the collective activities of interconnected neurons start to resemble a form of swarm intelligence [5], that is, it is possible for an agent (neuron) to communicate granules of knowledge to other agents (neurons) in its neighborhood rather than by means of the usual restricted model of movement of granules of information "upward" from neurons in one layer to neurons in a higher layer (e.g., in [1,6,8,19]).

Petri net models for two fundamental forms of rough neural computing are presented: training set production and AR schemes defined in the context of parameterized approximation spaces. The focus of the first form of rough neural computing is inductive learning and the production of training (optimal feature set selection) using knowledge reduction algorithms. This first form of neural computing can be important in designing neural networks defined in the context of parameterized approximation spaces. A high-level Petri net model of a neural network classifier with an optimal feature selection procedure in its front end is given. This model is followed by the development of a number of Petri net models of what are known as elementary approximation neurons (EA neurons). The design of a particular form of EA neuron is considered, namely, a neuron that includes an uncertainty (activation) function that constructs a granule approximation and a rough inclusion (threshold activation) function that measures the degree to which granule approximation is part of a target granule.

This chapter is organized as follows. An overview of set approximation, attribute reduction (derivation of minimal sets of attributes), decision rules, discretization, and rough membership set functions is presented in Sect. 2. The basic features of rough Petri nets are described in Sect. 3. A training set production model is briefly given in Sect. 4. A framework (scheme) for approximate reasoning in the context of parameterized approximation spaces is given in Sect. 5. A sample approximation space and an indistinguishability relation are also given in Sect. 5. Three elementary approximation neuron Petri net models are presented in Sect. 6. Also included in Sect. 6 is an elementary Petri net model for approximation neuron training.

2 Preliminaries

This section gives a brief overview of some fundamental concepts and features of rough set theory that are important to an understanding of the Petri net models given in this chapter.

2.1 Set Approximation

Rough set theory offers a systematic approach to set approximation [16]. To begin, let $S=(U, A)$ be an information system where U is a nonempty, finite set of objects

and A is a nonempty, finite set of attributes, where $a : U \to V_a$ for every $a \in A$. For each $B \subseteq A$, there is associated an equivalence relation $\mathrm{Ind}_A(B)$ such that

$$\mathrm{Ind}_A(B) = \{(x,x') \in U^2 \,|\, \forall a \in B \,.\, a(x) = a(x')\}.$$

If $(x, x') \in \mathrm{Ind}_A(B)$, we say that objects x and x' are indiscernible from each other relative to attributes from B. The notation $[x]_B$ denotes equivalence classes of $\mathrm{Ind}_A(B)$. Further, partition $U/\mathrm{Ind}_A(B)$ denotes the family of all equivalence classes of relation $\mathrm{Ind}_A(B)$ on U. For $X \subseteq U$, the set X can be approximated only from information contained in B by constructing a B-lower and a B-upper approximation denoted by $\underline{B}X$ and $\overline{B}X$, respectively, where $\underline{B}X = \{x \,|\, [x]_B \subseteq X\}$ and $\overline{B}X = \{x \,|\, [x]_B \cap X \neq \emptyset\}$.

2.2 Attribute Reduction and Decision Rules

An approach to finding a subset of attributes (reduct) with the same classificatory power as the entire set of attributes in an information system is briefly described in this section. This leads to a brief discussion about the derivation of decision rules with minimal descriptions in their left-hand sides. In deriving decision system rules, the discernibility matrix and discernibility function are essential. Given an information system S=(U, A), the $n \times n$ matrix (c_{ij}) called the discernibility matrix of S (denoted M_S) is defined in (1).

$$c_{ij} = \{a \in A \,|\, a(x_i) \neq a(x_j)\}, \quad \text{for } i, j = 1, \ldots, n. \tag{1}$$

A discernibility function f_S relative to discernibility matrix M_S for an information system S is a Boolean function of m Boolean variables $a_1^*, \ldots, a_m^*$ corresponding to attributes $a_1, \ldots, a_m$, respectively, and defined in (2).

$$f_S(a_1^*, \ldots, a_m^*) =_{df} \wedge \left\{\vee_{ij}^* | 1 \leqslant j \leqslant i \leqslant n, c_{ij} \neq \emptyset\right\}, \text{ where } c_{ij}^* = \left\{a^* | a \in c_{ij}\right\}. \tag{2}$$

Precise conditions for decision rules can be extracted from a discernibility matrix as in [30,31]. For the information system S, let $B \subseteq A$, and let $\wp(V_a)$ denote the power set of V_a, where V_a is the value set of a. For every $d \in A - B$, a decision function $d_d^B : U \to \wp(V_a)$ is defined in (3).

$$d_d^B(u) = \left\{v \in V_d \,|\, \exists u' \in U, \, (u',u) \in \mathrm{Ind}_A(B), d(u') = v\right\}. \tag{3}$$

In other words, $d_d^B(u)$ is the set of all elements of the decision column of S such that the corresponding object is a member of the same equivalence class as argument u. The next step is to determine a decision rule with a minimal number of descriptors on the left-hand side. Pairs (a, v), where $a \in A, v \in V$, are called *descriptors*. A decision rule over the set of attributes A and values V is an expression of the form given in (4).

$$a_{i_1}(u_i) = v_{i_1} \wedge \ldots \wedge a_{i_j}(u_i) = v_{i_j} \wedge \ldots \wedge a_{i_r}(u_i) = v_{i_r} \underset{S}{\Rightarrow} d(u_i) = v, \tag{4}$$

where $u_i \in U$, $v_{i_j} \in V_{a_{i_j}}$, $v \in V_d$, $j = 1, \ldots, r$ and $r \leq \text{card}(A)$. The fact that a rule is true is indicated by writing it in the form given in (5).

$$(a_{i_1} = v_{i_1}) \wedge \ldots \wedge (a_{i_r} = v_{i_r}) \underset{S}{\Rightarrow} (a_p = v_p). \tag{5}$$

The set of all prime implicants of f_S determines the set of all reducts of S [9]. A reduct is a minimal set of attributes $B \subseteq A$ that can be used to discern all objects obtainable by all of the attributes of an information system [37]. The set of all reducts of S is denoted by RED(S). A method used to find a proper subset of attributes of A with the same classificatory power as the entire set A has been termed *attribute reduction* [36].

Let $R \in$ RED(S) be a reduct in the set of all reducts in an information system S. For information system S, the set of decision rules constructed with respect to a reduct R is denoted OPT(S, R). Then the set of all decision rules derivable from reducts in RED(S) is the set in (6).

$$\text{OPT}(S) = \cup\{\text{OPT}(S,R) \,|\, R \in \text{RED}(S)\}. \tag{6}$$

2.3 Discretization

Suppose that we need to obtain approximate knowledge of a continuum (e.g., behavior of a sensor signal over an interval of time) by considering parts of the continuum. Discretization of a continuum entails partition a particular interval into subintervals of reals. For example, consider the interval of reals $V_a = [v_a, w_a]$ for values of an attribute $a \in A$ in a consistent decision system S=(U, A). Discretization of V_a entails searching for a partition P_a of V_a (i.e., discovering a partition of the value sets of attributes into intervals). In rough set theory, discretization leads to partitions of value sets so that, if the name of the interval containing an arbitrary object is substituted for any object in place of its original value in S, a consistent information system is also obtained.

2.4 Rough Membership Set Functions

In this section, we consider a set function form of the traditional rough membership function, which was introduced in [17].

Definition 1. Let S=(U, A) be an information system, $B \subseteq A$, and let $[u]_B$ be an equivalence class of an object $u \in U$ of $\text{Ind}_A(B)$. A set function $\mu_u^B : \wp(U) \rightarrow [0,1]$ is defined in (7).

$$\mu_u^B(X) = \frac{\text{card}(X \cap [u]_B)}{\text{card}([u]_B)}. \tag{7}$$

for any $X \in \wp(Y)$, $Y \subseteq U$, where $\wp(Y)$ denotes the power set of Y, called a *rough membership function* (rmf). A rough membership function provides a classification measure inasmuch as it tests the degree of overlap between the set X and an equivalence class $[u]_B$. The form of rough membership function in Definition 1 is slightly different from the classical definition [18] where the argument of the rough membership function is an object u and the set X is fixed. For example, let $X_{B_{\text{approx}}} \in \left\{ \bar{B}X, \underline{B}X \right\}$ denote a set approximation. Then we compute the degree of overlap between $X_{B_{\text{approx}}}$ and $[u]_B$ in (8).

$$\mu_u^B(X_{B_{\text{approx}}}) = \frac{\text{card}([u]_B \cap X_{B_{\text{approx}}})}{\text{card}([u]_B)}. \tag{8}$$

3 Rough Petri Nets

Rough Petri nets were introduced in [21] and elaborated in [22]. A rough Petri net models a process that implements one or more features of rough set theory. Rough Petri nets are derived from colored and hierarchical Petri nets as well as from rough set theory. Colored Petri nets provide a well-understood framework for introducing computational mechanisms (data types, variables, constants, and functions) that are useful in describing processes that carry out set approximation, information granulation, and engage in approximate reasoning. The new form of Petri net uses rough set theory, multivalued logic, and receptor process theory to extend colored Petri nets. Three extensions of colored Petri nets are described in this chapter: (1) multivalued guards, (2) receptor processes, and (3) rough computing. Briefly, this is explained as follows. Boolean valued guards in traditional colored Petri nets are replaced by multivalued guards in rough Petri nets. Let T be a set of transitions. In a colored Petri net, a guard is a mapping $G : T \to \{0, 1\}$. A transition is enabled if G returns 1. In keeping with an interest in modeling approximate reasoning, we augment colored Petri nets with guards of the form $G : T \to [0, 1]$. With this form of guard, it is possible to model a level-of-enabling of a transition that is more general than the usual "on/off" enabling model. Places in a Petri net represent the states of a system. An input place is a source of input for a method associated with a transition. An output place is a repository for results computed by a transition method. In a rough Petri net, an input place can be a receptor process. This form of input place responds to each stimulus from the environment by measuring a stimulus and by making each measurement available to a transition method. This extension of the input place convention in colored Petri nets is needed to model dynamically changing systems that perform actions in response to sensor signals. There is an iteration "inside" a receptor process that idealizes a typical sensor-action system found in agents, that is, the intent behind a provision of input places that are receptor processes is to model a sensor that enables a response mechanism represented by a transition method each time the sensor is stimulated. Rough computation is the third extension of colored Petri nets considered in this chapter. This feature is the hallmark of a rough Petri

net. It is characterized by the design of transition methods that compute rough set structures (e.g., attribute reduction) as well as values of measurable functions and rough integrals. This feature is important to us because we are interested in modeling parts of intelligent systems such as neurons in a neural processing module and information granulation. A rough Petri net also includes a strength-of-connection mapping from arcs to weights. This feature of rough Petri nets is useful in modeling classical neural computation. A rough Petri net provides a basis for modeling, simulating, and analyzing approximate reasoning (especially in the context of rough-neurocomputing), decision, and control systems. In what follows, it is assumed that the reader is familiar with classical Petri nets in Petri [25] and colored Petri nets in Jensen [7].

Definition 2. *Rough Petri Net*. A rough Petri net (rPN) is a structure $(\Sigma, P, T, A, N, C, G, E, I, W, \Re, \xi)$, where

- Σ is a finite set of nonempty data types called color sets.
- P is a finite set of places.
- T is a finite set of transitions.
- A is a finite set of arcs such that $P \cap T = P \cap A = T \cap A = \emptyset$.
- N is a 1–1 node function where $N : A \to (P \times T) \cup (T \times P)$.
- C is a color function where $C : P \to \Sigma$.
- G is a guard function where $G : T \to [0, 1]$.
- E is an arc expression function where $E : A \to Set\_of\_Expressions$ where $E(a)$ is an expression of type $C[p(a)]$ and $p(a)$ is the place component of $N(a)$.
- I is an initialization function where $I : P \to Set\_of\_Closed\_Expressions$ where $I(p)$ is an expression of type $C(p)$.
- W is a set of strengths-of-connections where $\xi : A \to W$.
- $\Re = \{\rho_\sigma \,|\, \rho_\sigma$ is a method that constructs a rough set structure or that computes a value$\}$.

A sample ρ_σ is a method that constructs a rough set structure (e.g., an upper approximation of a set X relative to a set of attributes B or the set OPT(S) of all rules derived from reducts of a decision system table for an information system S). Again, for example, ρ_σ can denote a rough membership function or a rough integral [17,22]. The availability of guards on transitions makes it possible to model sensor filters and various forms of fuzzy Petri nets. Higher order places representing receptor processes are part of rough Petri nets.

3.1 Receptor Processes

The notion of a receptor process comes from Dill [4]. In a rough Petri net, a receptor process is a higher order place that models a sensor. The input place labeled $?p1$ in Fig. 1(a), for example, represents a form of receptor process that accumulates a signal.

Definition 3. *Receptor Process.* A receptor process is a process that provides an interface between a system and its environment by recording its response to each stimulus in a finite set of sample sensor values (a signal) whenever stimuli are detected.

When an environment is a source of continuous stimulation, a receptor process continuously enqueues its responses and periodically enables any transition connected to it. The advantage in constructing such a Petri net model of sensor-dependent systems is that it facilitates reasoning about a system design and simulation of the responses of a system to stimuli. For example, let $?p1$ be a receptor process; X, a set of inputs (signal) produced by $?p1$, and let μ_u^e denote a rough membership function. Further, let $(Y, U-Y)$ be a partition defined by an expert e, and let $[u]_e$ denote a set in this partition containing u for a fixed $u \in U$ (see Fig. 1a).

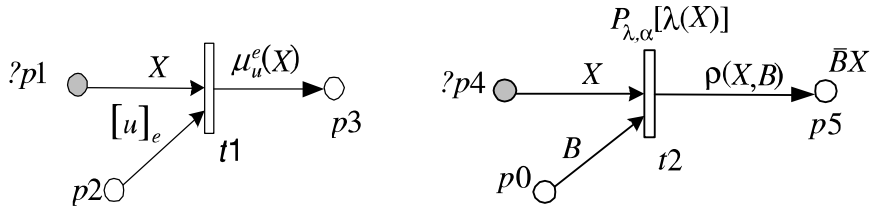

Fig. 1. (a) rPN without guard **(b)** guarded rPN

When transition $t1$ fires, $\mu_u^e(X)$ computes an rmf value. The label on transition $t2$ in Fig. 1b is an example of a Łukasiewicz guard (we explain this in this next section). Briefly, transition $t2$ is enabled, if $\alpha[\lambda(X)]$ holds for input X. The notation X denotes a set of values "produced" by the receptor process named $?p4$. The set X represents a signal or set of sample receptor process values that are accumulated over time. It is possible that $X = \{x\}$ (a restriction of X to a single sample receptor process value). In that case, the label x (rather than $\{x\}$) will be used to denote the input to a transition. The notation B on the arc from $p0$ to $t2$ represents a set of attributes that will be used by method ρ to construct a rough set structure. More details about receptor processes are given in [22].

3.2 Guarded Transitions

A guard $G(t)$ is an enabling condition associated with a transition t. In a rough Petri net, various families of guards can be defined that induce a level-of-enabling of transitions. Łukasiewicz guards were introduced in [24]. There are many forms of Łukasiewicz guards. Let $(L, \leq)$ be a lattice with a smallest element $\perp$ and with all other elements incomparable. In this chapter, $L - \{\perp\}$ is assumed to be equal to an interval of reals.

Definition 4. *Łukasiewicz Guard*. For a given function $\lambda(X)$ from the domain of variable X into $L - \{\perp\}$ and a condition α, i.e., a function from $L - \{\perp\}$ into $\{0, 1\}$, a Łukasiewicz guard $P_{\lambda,\alpha}(X)$ is a function from the domain of the variable X into L defined by

$$P_{\lambda,\alpha}(X) = \lambda(X), \text{ if } \alpha[\lambda(X)] = 1, \text{ and } \perp \text{ otherwise.}$$

We assume that a transition t labeled by $P_{\lambda,\alpha}(X)$ is enabled if and only if $\alpha[\lambda(X)]$, i.e., $\alpha[\lambda(X)] = 1$. The value of $\lambda(X)$ of $P_{\lambda,\alpha}(X)$ is a part of the output labeling the outgoing edges from t, if t is fired. Examples of conditions α are $0 < \lambda(X) \leq 1$ or $\lambda(X) \geq b > 0$ where b is a selected b in (0, 1].

4 Training Set Production

Training set production has been used by a number of researchers to reduce the size of the data set needed to train a neural network [32,38,39]. This approach entails finding reducts (attribute reduction), i.e., finding minimum sets of attributes that have the same classificatory power as the original set of attributes in an information system. A reduct computation has been described as a pure feature selection procedure [38].

Algorithm:
Basic signal classification based on selected reduct

Given:
Set of n-case stimuli (signals) containing an n-dimensional pattern with k real-valued attributes (sensors) A of the form $(a_1(x_i) \ldots a_k(x_i))$ for $a_i \in A,\ x_i \in X$.

1. For each $(a_1(x_i) \ldots a_k(x_i))$, determine decision value $v_d \in V_d$ (set of decision values).
2. Construct decision table $S_d = (X, A \cup \{d\})$, where each row of S_d is of the form $(a(x_i) \ldots a(x_i)\, v_d)$.
3. Compute RED(S_d) from S_d.
4. Select reduct $R \in$ RED(S_d). Assume that R has m attributes.
5. Compose reduced decision system $S_d = (X, A \backslash R \cup \{d\})$, where R is treated as a set of features describing all concepts in S_d.
6. Construct error back-propagation neural network (bpNN) over the reduced data set.
7. Train and test neural network bpNN.
8. Repeat steps 4 to 7 until the best classification result is obtained.

The notation *con*( ) in Fig. 2 denotes a procedure that performs signal conditioning. A sample form of *con*( ) based on a sensor fusion model using a discrete rough integral is given in [22].

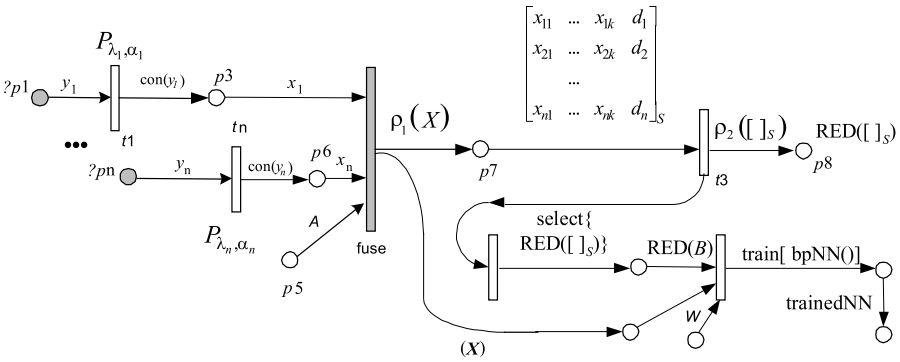

Fig. 2. Reduct-based neural net classifier model

5 Granular Computing Model

A brief introduction to a rough-neurocomputing Petri net model based on an adaptive calculus of granules is given in this section. Information granule construction and parameterized approximation spaces provide the foundation for the model of rough-neurocomputing [29]. A fundamental feature of this model is the design of neurons that engage in knowledge discovery. Mechanically, such a neuron returns granules (synthesized knowledge) derived from input granules. It has been pointed out that there is an analogy between calculi of granules in distributed systems and rough neural computing [29]:

1. An agent with input and output ports providing communication links with other agents provides a model for a neuron η (analogously agent *ag*) with inputs supplied by neurons $\eta_1, \ldots, \eta_k$ (analogously agents $ag_1, \ldots, ag_k$), responds with output by η, and η is designed with a parameterized family of activation functions represented as rough connectives. In effect, a neuron resembles the model of an agent proposed by Milner [13].
2. Values of rough inclusions are analogous to weights in traditional neural networks.
3. Learning in a system governed by an adaptive calculus of granules is in the form of back propagation where incoming signals are assigned a proper scheme (granule construction) and a proper set of weights in negotiation and cooperation with other neurons.

5.1 Parameterized Approximation Spaces

A step toward the realization of an adaptive granule calculus in a rough-neurocomputing scheme is described in this section and is based on [31]. In a scheme for infor-

mation granule construction in a distributed system of cooperating agents, weights are defined by approximation spaces. In effect, each agent (neuron) in such a scheme controls a local parameterized approximation space.

Definition 5. *Parameterized Approximation Space*. A parameterized approximation space is a system $AS_{\#,\$} = (U, I_{\#}, R, \nu_{\$})$ where #, \$ denote vectors of parameters; U is a nonempty set of objects;

- $I_{\#} : U \to \wp(U)$ is an uncertainty function where $\wp(U)$ denotes the power set of U;
- $R \subseteq \wp(U)$ is a family of patterns; and
- $\nu_{\$} : \wp(U) \times \wp(U) \to [0, 1]$ denotes rough inclusion.

The uncertainty function defines a set of similarly described objects for every object x in U. A constructive definition of an uncertainty function can be based on the assumption that some metrics (distances) are given on attribute values. The family R describes a set of patterns (e.g., representing the sets described by the left-hand sides of decision rules). A set $X \subseteq U$ is definable on $AS_{\#,\$}$ if it is a union of some values of the uncertainty function. The rough inclusion function $\nu_{\$}$ defines the value of inclusion between two subsets of U. Using rough inclusion, the neighborhood $I_{\#}(x)$ can usually be defined as a collection of close objects. Also note that for some problems it is convenient to define an uncertainty set function of the form $I_{\#} : \wp(U) \to \wp(U)$. This form of uncertainty function works well in signal analysis, where we want to consider a domain over sets of sample signal values.

5.2 Sample Approximation Space

A very simple sample parameterized approximation space is given in this section.

Example 1. Threshold-Based Approximation Space This example is derived from [35]. Consider the information system S =(U,A). Let $A = \{a\}$, where the attribute is real-valued, and let U be a nonempty set of reals with $x \in U$. Consider two elementary granules $a(x) \in [v_1, v_2]$ and $a(x) \in [v'_1, v'_2]$ for two ordinate sets of points defined with real numbers v_1, v_2, v'_1, v'_2 where $v_1 < v_2$ and $v'_1 < v'_2$. For simplicity, assume that the subinterval $[v_1, v_2]$ is contained in the interval $[v_1, v'_2]$. We want to measure the degree of inclusion of granule $a(x) \in [v_1, v_2]$ in granule $a(x) \in [v'_1, v'_2]$ (i.e., we assume that elements of R and neighborhoods of objects are such intervals). First, we introduce an overlapping range function r_a to measure the overlap between a pair of subintervals:

$$r_a\left([v_1, v_2], [v'_1, v'_2]\right) = \max\left(\left\{\min\left(\left\{v_2, v'_2\right\}\right) - \max\left(\left\{v_1, v'_1\right\}\right), 0\right\}\right).$$

The uncertainty function $I_{B,\delta} : U \to \wp(U)$ is defined as follows:

$$I_{B,\delta}(x) = \begin{cases} [v_1, v_2], \text{ if } \lfloor a(x)/\delta \rfloor \in [v_1, v_2] \\ [v_1', v_2'], \text{ if } \lfloor a(x)/\delta \rfloor \in [v_1', v_2'] \\ [0,0], \text{ otherwise,} \end{cases}$$

where $\lfloor a(x)/\delta \rfloor$ denotes the greatest integer less than or equal to $a(x)/\delta$. A rough inclusion function $\nu_{t_a} : \wp(U) \times \wp(U) \to [1, 0]$ is then defined as follows:

$$\nu_{t_a}\left[I_{B,\delta}(x), I_{B,\delta}(x')\right] = \frac{r_a\left[I_{B,\delta}(x), I_{B,\delta}(x')\right]}{v_2 - v_1} \in [0, t_a) \cup [t_a, 1].$$

Elementary granule $a(x) \in [v_1, v_2]$ is included in elementary granule $a(x) \in [v_1', v_2']$ in degree t_a if and only if

$$\frac{r_a\left[I_{B,\delta}(x), I_{B,\delta}(x')\right]}{v_2 - v_1} \geq t_a \qquad \text{(threshold criterion).}$$

This version of a parameterized approximation space depends on a threshold parameter t_a used as a criterion for the degree of inclusion of one interval in another interval at or above the threshold t_a whenever $\nu_{t_a}\left[I_{B,\delta}(x), I_{B,\delta}(x')\right] \geq t_a$. Changes in parameter δ in the uncertainty function $I_{B,\delta}$ and threshold parameter t_a in the inclusion model ν_{t_a} in this sample approximation space will change the result. With a threshold t_a, ν_{t_a} acts like a high-pass filter (it "passes" high values of v above t_a and rejects low values of v). The values of these parameters can be learned during training (i.e., during training, adjustments in the parameters are made to improve the classification of elementary input granules).

5.3 Indistinguishability Relation

To begin, let $S = (U, A)$ be an infinite information system where U is a nonempty subset of the reals $\Re$ and A is a nonempty, finite set of attributes, where $a : U \to V_a$ for every $a \in A$. Let $a(x) \geq 0, \delta > 0, x \in \Re$ (set of reals), and let $\lfloor a(x)/\delta \rfloor$ denote the greatest integer less than or equal to $a(x)/\delta$ ["floor" of $a(x)/\delta$] for attribute a. If $a(x) < 0$, then $\lfloor a(x)/\delta \rfloor = -\lfloor |a(x)|/\delta \rfloor$. The parameter δ serves as a "neighborhood" size on real-valued intervals. Reals within the same subinterval bounded by $k\delta$ and $(k+1)\delta$ are considered indistinguishable. For each $B \subseteq A$, there is associated an equivalence relation $\mathrm{Ing}_{A,\delta}(B)$ defined in (9).

$$\mathrm{Ing}_{A,\delta}(B) = \left\{ (x, x') \in \Re^2 \,|\, \forall a \in B.\ \lfloor a(x)/\delta \rfloor = \lfloor a(x')/\delta \rfloor \right\}. \tag{9}$$

If $(x, x') \in \mathrm{Ing}_{A,\delta}(B)$, we say that objects x and x' are indistinguishable from each other relative to attributes from B. A subscript Id denotes a set of identity sensors

$Id(x) = x$. They are introduced to avoid a situation, where there is more then one stimulus for which a sensor takes the same value. From (9), we can write

$$\mathrm{Ing}_{A,\delta}(B \cup Id) = \left\{ \begin{array}{l} (x,x') \in \Re^2 | \, \lfloor x/\delta \rfloor = \lfloor x'/\delta \rfloor \ \wedge \\ \forall a \in B. \ \lfloor a(x)/\delta \rfloor = \lfloor a(x')/\delta \rfloor \end{array} \right\}.$$

The notation $[x]_B^\delta$ denotes equivalence classes of $\mathrm{Ing}_{A,\delta}(B)$. Further, partition $U/\mathrm{Ing}_{A,\delta}(B)$ denotes the family of all equivalence classes of relation $\mathrm{Ing}_{A,\delta}(B)$ on U. For $X \subseteq U$, the set X can be approximated only from information contained in B by constructing a B-lower and a B-upper approximation denoted by $\underline{B}X$ and $\overline{B}X$, respectively, where $\underline{B}X = \left\{ x \mid [x]_{B \cup Id}^\delta \subseteq X \right\}$ and $\overline{B}X = \left\{ x \mid [x]_{B \cup Id}^\delta \cap X \neq \emptyset \right\}$. In some cases, we find it necessary to use a sensor reading y (an ordinate or "vertical" value) instead of stimulus x. In such cases, we create an equivalence class consisting of all points (ordinate values) for which sensor readings are "close" to y and define $[y]_B^\delta = \left\{ x \in \Re | \ \forall a \in B. \ \left\lfloor \frac{a(x)}{2\delta} \right\rfloor = \lfloor \frac{y}{2\delta} \rfloor \right\}$, where $[y]_B^\delta$ consists of equivalence classes $[x]_B^\delta$ such that $a(x) = y$. This is quite important when we want to extract information granules relative to sensor signals (sensor measurements rather than sensor stimuli then hold our attention).

Proposition 1. The relation $\mathrm{Ing}_{A,\delta}(B)$ is an equivalence relation.

Note that it is also possible to define upper and lower approximations of a set in the context of the uncertainty function and rough inclusion in an approximation space, that is, we define an uncertainty function $I_\# : U \to \wp(U)$ and rough inclusion $\mu : X \times Y \to [0, 1]$. Then we obtain

$$\mathrm{LOW}\left(AS_{\#,\$}, X\right) = \{x \in U \,|\, v_\$ \left[I_\#(x), X\right] = 1\},$$
$$\mathrm{UPP}\left(AS_{\#,\$}, X\right) = \{x \in U \,|\, v_\$ \left[I_\#(x), X\right] > 0\}.$$

This is the approach taken in [29]. There is some flexibility in the way we define the uncertainty function for an approximation space. For some problems, it is more convenient to define a set function of the form $I_\# : \wp(U) \to \wp(U)$. This form of uncertainty function works well in signal analysis, where we want to consider a domain over sets of sample signal values. This change in $I_\#$ leads to slightly different definitions of upper and lower approximations in an approximation space:

$$\mathrm{LOW}\left(AS_{\#,\$}, Y\right) = \{X \in \wp(U) \,|\, v_\$ \left[I_\#(X), Y\right] = 1\},$$
$$\mathrm{UPP}\left(AS_{\#,\$}, Y\right) = \{X \in \wp(U) \,|\, v_\$ \left[I_\#(X), Y\right] > 0\}.$$

5.4 More Sample Approximation Spaces

In this section, we consider a parameterized approximation space defined relative to the indistinguishability relation and uncertainty set functions where the domain of such a function is the power set (set of subsets) of U.

Example 2. Indistinguishability-Based Approximation Space. Consider a parameterized approximation space with an uncertainty set function $I_{B,\delta}$ that constructs a granule (namely, an upper approximation) based on knowledge of B and indistinguishability relation with parameter δ and a rough inclusion function v with threshold parameter t_0. We write $v_{t_0}(X,Y)$ to denote $v(X,Y) > t_0$. For simplicity, the traditional $\overline{BX}$ is constructed by a method named constructApprox as part of the definition of $I_{B,\delta}$, where $I_{B,\delta} : \wp(U) \to \wp(U)$ for $I_{B,\delta}(X) = \text{constructUPP}(X) = \overline{BX}$, and let $Y = [y]_B^\delta$. In this example, rough inclusion is defined as follows:

$$v : \wp(U) \times \wp(U) \to [0,1] \text{ for } v[I_{B,\delta}(X),Y] = \frac{\rho\left[I_{B,\delta}(X) \cap Y\right]}{\rho(Y)},$$

where

$$v(X,Y) = \frac{\rho\left(\overline{BX} \cap [y]_B^\delta\right)}{\rho\left([y]_B^\delta\right)} = \frac{\int_{\overline{BX} \cap [y]_B^\delta} 1 \, dx}{\int_{[y]_B^\delta} 1 \, dx}.$$

The rough inclusion of granule $I_{B,\delta}(X)$ is acceptable in granule Y, provided that the following constraint is satisfied:

$$v_{t_0}(X,Y), \text{ i.e., } v(X,Y) > t_0.$$

The essential thing to notice about this variant of $I_{B,\delta}$ in this example is that it constructs the granule $\overline{BX}$ from its domain X. The rough inclusion function then measures the degree of overlap between $\overline{BX}$ and a set represented by Y. The composition of the set Y is not treated in this example. The parameters for $I_{B,\delta}$ are δ (tolerance) and set of attributes (features) B. The parameter for v is the threshold t_a.

6 Approximation Neuron Models

The parameters in a parameterized approximation space can be treated as an analogy to neural network weights, and each instance of such a granule-producing agent with a parameterized approximation space design parallels the architecture of neurons in a conventional neural network (see Fig. 1 in Chap. 2). To carry this analogy a step further, the parameters of an approximation space should be learned to induce the relevant information granules.

A neuron with a parameterized approximation space in its design is called an approximation neuron. In its simplest form, such a neuron constructs an elementary granule as a result of approximating the information received from its inputs. In more elaborate forms of an approximation neuron, for example, the output of the neuron may take the form of a rule derived from a condition vector of inputs, a reduct derived from a received decision table, or a set of rules derived from a received reduct and received decision table. The particular configuration of such a

neuron depends on instantiation of the approximation space and particular activation functions used in designing the neuron. The design of an approximation neuron changes each time we modify the definition of the uncertainty function $I_{\#}$ and rough inclusion function $v_{\$}$ as well the parameters in # and $ chosen for these functions. In this section, we consider two fairly basic models of elementary approximation neurons (EA neurons). An EA neuron is a neuron that constructs an elementary granule as its output. An elementary information granule is an information granule that contains a single "piece" of information (e.g., an attribute, or sensor value, a condition for a rule, a measurement of rough inclusion). The output of such an approximation neuron is a rough inclusion value.

6.1 Threshold-Based Approximation Neuron

The approximation space in Example 1 suggests the possibility of designing a simple prototype neural network where changes in the parameters rather than changes in weights provide a basis for training. It is possible to design a simple prototype neural network where changes in the parameters rather than changes in weights provide a basis for training in the context of a parameterized approximation space. We want to consider a threshold-based approximation neuron with an elementary granule as its output, namely, $v\left[I_{B,\delta}(x), I_{B,\delta}(x')\right]$ (see Fig. 3). In Fig. 3, a Petri net is given to model an approximation neuron. This is an example of a rough Petri net. The label $?p_0$ in Fig. 3 denotes a receptor process (always input ready) "connected" to the environment of an agent.

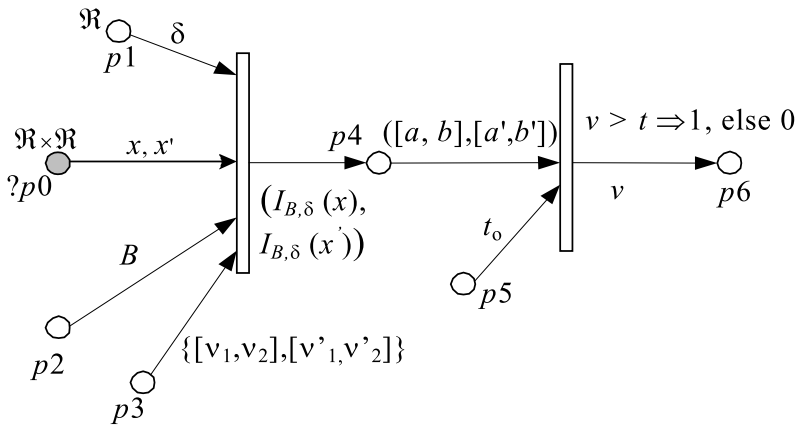

Fig. 3. Threshold-based neuron design

6.2 Indistinguishability-Based Approximation Neuron

It is also possible to design a prototype neural network based on the indistinguishability relation. In the form of neural computation described in this section, training entails changing δ (the interval width) until the rough inclusion function value exceeds the threshold t_0. An indistinguishability-based approximation neuron can be designed with an elementary granule as its output, namely, $v\left[I_{B,\delta}(X), Y\right]$, where $I_{B,\delta}(X)$ computes $\underline{B}X$, and on the output we have 1 if card($\underline{B}X \cap Y$)/card(Y) $>t_0$ and 0 otherwise (see Fig. 4).

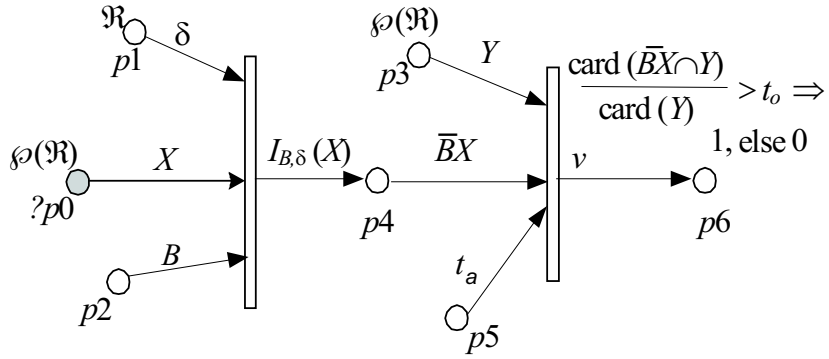

Fig. 4. Distinguishability-based neuron design

6.3 Approximation Neuron Training Model

In this section, a model for training is limited to a single approximation neuron. Up to this point, no guarded transitions have been used in Petri net models of approximation neurons. In this section, we consider a Petri net approximation neuron training model with three transitions, *neuron*, *train*, and a third transition named *approx* (see Fig. 5). Except for the guards and one extra communication transition, the Petri net in Fig. 3 is represented in Fig. 5 by the *neuron* transition together with its inputs and output. The firing of a *neuron* results in the computation of the rough inclusion of input X relative to relative to some set Y and either an initial value of δ or a changed value of δ "propagated back" from the transition train to place $p1$. Changes in δ occur during training and are the result of executing a procedure called *BP* (see Fig. 5). The term *back propagation* is typically used to describe training a multilayer perceptron using a gradient descent applied to a sum-of-squares error function. Training in the basic neuron in Fig. 5 is much simpler since we are dealing only with the need to modify one parameter δ. If the transition train in Fig. 5 had more than one rough inclusion computation as input and more than one δ to adjust, then it would be appropriate to consider some form of traditional back-propagation method

in adjusting the δ values. The transition *train* is enabled if $\nu[I_{B,\delta}(X),Y]-t_0<0$. In other words, this transition is enabled whenever $\nu[I_{B,\delta}(X),Y]<t_0$ (i.e., rough inclusion falls below threshold t_0). Each time the transition *train* fires, a new δ value is computed by the error function $\mathrm{BP}_\nu(\delta)$. The output place labeled $p1$ for the transition *train* in Fig. 5 is an alias for the input place $p1$ for the transition *neuron*. In the simple neural training model in Fig. 5, what happens to the neuron output when the rough inclusion value falls in the interval $[t_0,1]$ has been modeled with transition approx, which is enabled when the neuron output is at or above the required threshold. Transition approx serves as a high-pass filter, and only transmits *v*-granules in $[t_0,1]$ to other agents.

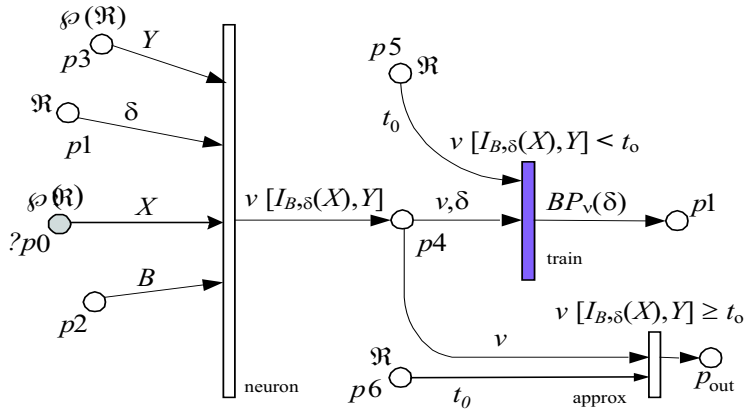

Fig. 5. Basic neuron training model

7 Conclusion

An approach to designing neurons using parameterized approximation spaces has been given in this chapter. In rough-neurocomputing, an AR scheme is defined in the context of one or more parameterized approximation spaces (denoted by $AS_{\#,\$}$). The methodology of an AR scheme is revealed by defining the universe U and the particular uncertainty and rough inclusion functions in the space $AS_{\#,\$}$. The scope of the AR scheme for a rough neural network is determined by identifying the particular universe U, the set of attributes reflecting the local knowledge of an agent (neuron), and the specific parameters in the vectors #, $. A parameterized approximation space for an AR scheme is defined in the context of traditional rough set theory (granule approximation) as well as the notion of being part of in a degree from approximate rough mereology. Training a rough neural network entails fine-tuning the parameters of its approximations spaces to make it possible for the network to achieve optimal classificatory power. A trainable AR scheme that consists of a single parameterized approximation space that constructs an information granule that

approximates its inputs, outputs some form of information granule, and adjusts its parameters to reduce the error in its output is called an approximation neuron. In this chapter, Petri net models for what are known as elementary approximation neurons have been given. An important consideration not included in the approximation neuron models but treated independently is given in this chapter, namely, attribute reduction. It is well known that performance improvement results from the inclusion of a feature subset selection procedure (finding a minimum-length reduct) in the front end of a traditional back-propagation neural network classification system. A Petri net model of this form of neural network has also been given.

Acknowledgments

The research of James Peters and Sheela Ramanna has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) research grant 185986 and research grant 194376, respectively. The research of Zbigniew Suraj was partially supported by grant #8T11C02519 from the State Committee for Scientific Research (KBN) in Poland. The research of Maciej Borkowski was supported by Manitoba Hydro.

References

1. M.A. Arbib. The artificial neuron. In E. Fiesler, R. Beale, editors, *Handbook of Neural Computation*, B1.7, Institute of Physics, Bristol, 1997.
2. I.M. Bochenski. *A History of Formal Logic*. Chelsea, New York, 1956.
3. B. Chakraborty. Feature subset selection by neuro-rough hybridization. In W. Ziarko, Y. Yao, editors, *Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing (RSCTC 2000)*, LNAI 2005, 519–526, Springer, Berlin, 2001.
4. D.L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits*. MIT Press, Cambridge MA, 1989.
5. Bonabeau et al. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Oxford, 2000.
6. L. Han, J.F. Peters, S. Ramanna, R. Zhai. Classifying faults in high voltage power systems: A rough-fuzzy neural computational approach. In N. Zhong, A. Skowron, S. Ohsuga, editors, *New Directions in Rough Sets, Data Mining, and Granular-Soft Computing*, LNAI 1711, 47–54, Springer, Berlin, 1999.
7. K. Jensen. *Coloured Petri Nets—Basic Concepts, Analysis Methods and Practical Use*, Vol. 1. Springer, Berlin, 1992.
8. T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78: 1464–1480, 1999.
9. J. Komorowski, Z. Pawlak, L. Polkowski, A. Skowron. Rough sets: A tutorial. In S.K. Pal, A. Skowron, editors, *Rough Fuzzy Hybridization: A New Trend in Decision-Making*, 3–98, Springer, Singapore, 1999.
10. P.J. Lingras. Comparison of neofuzzy and rough neural networks. *Information Sciences*, 110: 207–215, 1998.

11. P.J. Lingras. Fuzzy-rough and rough-fuzzy serial combinations in neurocomputing. *Neurocomputing: An International Journal*, 36: 29–44, 2001.
12. R. Milner. *Calculus of Communicating Systems*. Report number ECS-LFCS-86-7 of Computer Science Department, University of Edinburgh, Edinburgh, 1986.
13. R. Milner. *Communication and Concurrency*. Prentice-Hall, New York, 1989.
14. S. Mitra, P. Mitra, S.K. Pal. Evolutionary modular design of rough knowledge-based network with fuzzy attributes. *Neurocomputing: An International Journal*, 36: 45–66, 2001.
15. S. K. Pal, S. Mitra. Multi-layer perceptron, fuzzy sets and classification. *IEEE Transactions on Neural Networks*, 3: 683–697, 1992.
16. Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning About Data*. Kluwer, Dordrecht, 1991.
17. Z. Pawlak, J.F. Peters, A. Skowron, Z. Suraj, S. Ramanna, M. Borkowski. Rough measures: Theory and applications. In S. Hirano, M. Inuiguchi, S. Tsumoto, editors, *Rough Set Theory and Granular Computing*, Vol. 5(1/2) of Bulletin of the International Rough Set Society, 177–184, 2001.
18. Z. Pawlak, A. Skowron. Rough membership functions. In R. Yager, M. Fedrizzi, J. Kacprzyk, editors, *Advances in the Dempster–Shafer Theory of Evidence*, 251–271, Wiley, New York, 1994.
19. W. Pedrycz, L. Han, J.F. Peters, S. Ramanna, R. Zhai. Calibration of software quality: Fuzzy neural and rough neural computing approaches. *Neurocomputing: An International Journal*, 36: 149–170, 2001.
20. J.F. Peters, L. Han, S. Ramanna. Rough neural computing in signal analysis. *Computational Intelligence*, 17(3): 493–513, 2001.
21. J.F. Peters, S. Ramanna. A rough set approach to assessing software quality: Concepts and rough Petri net model. In S.K. Pal, A. Skowron, editors, *Rough Fuzzy Hybridization: A New Trend in Decision-Making*, 349–380, Springer, Berlin, 1999.
22. J.F. Peters, S. Ramanna, M. Borkowski, A. Skowron, Z. Suraj. Sensor, filter and fusion models with rough Petri nets. *Fundamenta Informaticae*, 47: 307–323, 2001.
23. A. Skowron, J. Stepaniuk. Information granule decomposition. *Fundamenta Informaticae*, 47(3/4): 337–350, 2001.
24. J.F. Peters, A. Skowron, Z. Suraj, S. Ramanna. Guarded transitions in rough Petri nets. In *Proceedings of the 7th European Congress on Intelligent Systems and Soft Computing (EUFIT'99)*, 203–212, Verlag Mainz, Aachen, 1999.
25. C.A. Petri. *Kommunikation mit Automaten*. Institut für Instrumentelle Mathematik, Schriften des IIM Nr.3, Bonn, 1962.
26. L. Polkowski, A. Skowron. Rough mereological calculi of granules: A rough set approach to computation. *Computational Intelligence*, 17(3): 472–492, 2001.
27. L. Polkowski, A. Skowron. Rough mereology. In *Proceedings of the International Symposium on Methodology for Intelligent Systems (ISMIS'94)*, LNAI 869, 85–94, Springer, Berlin, 1994.
28. L. Polkowski, A. Skowron. Rough-neuro computing. In W. Ziarko and Y. Yao, editors, *Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing (RSCTC 2000)*, LNAI 2005, 57–64, Springer, Berlin, 2001.
29. A. Skowron. Toward intelligent systems: Calculi of information granules. *Bulletin of the International Rough Set Society*, 5: 9–30, 2001.
30. A. Skowron, C. Rauszer. The discernibility matrices and functions in information systems. In R. Słowiński, editor, *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, 331–362, Kluwer, Dordrecht, 1992.

31. A. Skowron, J. Stepaniuk. Decision rules based on discernibility matrices and decision matrices. In *Proceedings of the 3rd International Workshop on Rough Sets and Soft Computing (RSSC'94)*, 602–609, San Jose, CA, 1994.
32. A. Skowron, J. Stepaniuk. Information granules in distributed environment. In N. Zhong, A. Skowron, S. Ohsuga, editors, *Proceedings of the 3rd International Workshop on Rough Sets, Data Mining, and Granular-Soft Computing (RSFDGrC'99)*, LNAI 1711, 357–365, Springer, Berlin, 1999.
33. A. Skowron, J. Stepaniuk. Information granules: Towards foundations of granular computing. *International Journal of Intelligent Systems*, 16: 57–1044, 2001.
34. A. Skowron, J. Stepaniuk, J.F. Peters. Extracting patterns using information granules. *Bulletin International Rough Set Society* 5: 135–142, 2001.
35. A. Skowron, J. Stepaniuk, J.F. Peters. Hierarchy of information granules. In L. Czaja, editor, *Proceedings of the Workshop on Concurrency, Specification and Programming (CSP 2001)*, 254–268, Warsaw, Poland, 2001.
36. R.W. Swiniarski. *RoughNeuralLab, software package*. Developed at San Diego State University, San Diego, CA, 1995.
37. R.W. Swiniarski, L. Hargis. Rough sets as a front end of neural networks texture classifiers. *Neurocomputing: An International Journal*, 36: 85–103, 2001.
38. R.W. Swiniarski, A. Skowron. Rough set methods in feature selection and recognition. *Pattern Recognition Letters*, 24(6): 833–849, 2003.
39. M.S. Szczuka. Rough sets and artificial neural networks. In L. Polkowski, A. Skowron, editors, *Rough Sets in Knowledge Discovery 2: Applications, Cases Studies and Software Systems*, 449–470, Physica, Heidelberg, 1998.
40. P. Wojdyłło. Wavelets, rough sets and artificial neural networks in EEG analysis. In *Proceedings of the 1st International Conference on Rough Sets and Current Trends in Computing (RSCTC'98)*, LNAI 1424, 444–449, Springer, Berlin, 1998.
41. L.A. Zadeh. Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, 4: 103–111, 1996.
42. L.A. Zadeh. The key roles of information granulation and fuzzy logic in human reasoning, concept formulation and computing with words. In *Proceedings of the 5th International Conference on Fuzzy Systems (FUZZ-IEEE'96)*, 8–11, New Orleans, LA, 1996.
43. N. Zhong, A. Skowron, S. Ohsuga, editors. *Proceedings of the 3rd International Workshop on New Directions in Rough Sets, Data Mining, and Granular Soft Computing (RSFDGrC'99)*, LNAI 1711, Springer, Berlin, 1999.
44. W. Ziarko, Y.Y. Yao , editors. *Proceedings of the International Conference on Rough Sets and Current Trends in Computing (RSCTC 2000)*, LNAI 2005, Springer, Berlin, 2001.