Chapter 2
Rough-Neurocomputing: An Introduction

Sankar K. Pal,[1] James F. Peters,[2] Lech Polkowski,[3] Andrzej Skowron[4]

[1] Machine Intelligence Unit, Indian Statistical Institute, 203 B.T. Road, Kolkata 700108, India
sankar@isical.ac.in
[2] Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, Manitoba R3T 5V6, Canada
jfpeters@ee.umanitoba.ca
[3] Polish-Japanese Institute of Information Technology Research Center, Koszykowa 86, 02-008 Warsaw, Poland
and
Department of Mathematics and Computer Science, University of Warmia and Mazury, Żołnierska 14a, 10-561 Olsztyn, Poland
polkow@pjwstk.edu.pl
[4] Institute of Mathematics, Warsaw University, Banacha 2, 02-097 Warsaw, Poland
skowron@mimuw.edu.pl

Summary. This chapter presents a new paradigm for neurocomputing that has its roots in rough set theory. Historically, this paradigm has three main threads: production of a training set description, calculus of granules, and interval analysis. This paradigm gains its inspiration from the work of Pawlak on rough set philosophy as a basis for machine learning and from work on data mining and pattern recognition by Swiniarski and others in the early 1990s. The focus of this work is on the production of a training set description and inductive learning using knowledge reduction algorithms. This first thread in rough-neurocomputing has a strong presence in current neurocomputing research. The second thread in rough-neurocomputing has two main components: information granule construction in distributed systems of agents and local parameterized approximation spaces (see Sect. 2.2 and Chap. 3). A formal treatment of the hierarchy of relations of being a part to a degree (also known as approximate *rough mereology*) was introduced by Polkowski and Skowron in the mid- and late-1990s. Approximate rough mereology provides a basis for an agent-based, adaptive calculus of granules. This calculus serves as a guide in designing rough-neurocomputing systems. A number of touchstones of rough-neurocomputing have emerged from efforts to establish the foundations for granular computing: cooperating agent, granule, granule measures (e.g., inclusion, closeness), and approximation space parameter calibration. The notion of a cooperating agent in a distributed system of agents provides a model for a neuron. Information granulation and granule approximation define two principal activities of a neuron. Included in the toolbox of an agent (neuron) are measures of granule inclusion and closeness of granules. Agents (neurons) acquire knowledge by granulating (fusing) and approximating sensor inputs and input (granules) from other agents. The second component of the granular form of rough-neurocomputing is a new approach to training agents (neurons). In this new paradigm, training a network of agents (neurons) is defined by algorithms for adjusting parameters in the parameter space of each agent. Parameters accessible to rough neurons replace the usual scalar weights on (strengths-of-) connections between neurons. Hence, learning in a rough neural

network is defined relative to local parameter adjustments. In sum, the granule construction paradigm provides a model for approximate reasoning by systems of communicating agents. The third thread in rough-neurocomputing stems from the introduction of a rough set approach to interval analysis by Banerjee, Lingras, Mitra, and Pal in the later part of the 1990s. This work has led to a variety of new rough-neurocomputing computational models. This chapter gives a brief presentation of an agent (neuron)-based calculus of granules. The design of different kinds of rough neurons is considered. Architectures of a number of different rough-neurocomputing schemes are also considered.

1 Introduction

The hint that rough set theory provides a good basis for neuro-computing can be found in a discussion about machine learning by Zdzislaw Pawlak in 1991 [26]. Inductive learning is divided into two phases that are reminiscent of training in classical neural computing: closed-world training and open-world training. Closed-world training focuses on constructing a pair (R_0, U_0), where R_0 is an initial set of classification rules and U_0 is an initial set of classified objects (initial universe). For each object $x \in U_0$, an agent is able to classify x based on identified features of the object (e.g., color, shape, weight, velocity). The aim of open-world training is to achieve complete knowledge of the universe by constructing (R_c, U_c), where R_c is created either as an initial set of classification rules or by modifying old rules and U_c is a complete set of classified objects (complete universe). A particular condition vector of feature values provides the basis for a decision in classifying an object in the set U_c. To some extent, this form of training is analogous to selecting a training set used to calibrate a neural network. During open-world training, an agent attempts to use R_0 to classify further (possibly new) objects by finding the condition vector in R_0 that most closely matches the experimental condition vector for a new object. In effect, the condition vectors in R_0 provide a "codebook" to define the space of input patterns. The trick is to use the codebook to identify the feature pattern of each new object. When an object x cannot be classified using R_0, a new classification rule $\chi \Rightarrow \mathrm{d}_\chi$ is formulated, R_c is augmented to reflect the knowledge about the changing universe (i.e., $R_c = R_0 \cup \{\chi \Rightarrow d_\chi\}$), and U_0 is augmented with newly classified objects $(U_c = U_0 \cup \{x\})$. The inductive learning method resembles learning vector quantization and self-organizing feature maps described in [1, 11]. The approach outlined above is simplified. For more advanced discussions on the rough set approach to inductive learning, refer to Chaps. 3 and 25.

The studies of neural networks in the context of rough sets [2, 5, 10, 14, 19, 21, 22, 24, 35–37, 41, 57, 60–63, 69, 74] and granular computing [12, 31, 40, 44–46, 48, 52, 53, 55, 74] are extensive. An intuitive formulation of information granulation was introduced by Zadeh [70, 71]. Practical applications of rough-neurocomputing have recently been found in predicting urban highway traffic volume [14], speech analysis [19, 24], classifying the waveforms of power system faults [10], signal analysis [36], assessing software quality [31], and control of autonomous vehicles [20]. In its most general form, rough-neurocomputing provides a basis for granular

computing. A rough mereological approach to rough neural networks springs from an interest in knowledge synthesized (induced) from successive granule approximations performed by neurons (cooperating agents) [44]. The distributed agent model for a neural network leads naturally to nonlayered neural network architectures, that is, it is possible for an agent (neuron) to communicate granules of knowledge to other agents (neurons) in its neighborhood rather than following the usual restricted model of a movement of granules "upward" from neurons in one layer to neurons in a higher layer. For this reason, the distributed agent model for rough-neurocomputing is reminiscent of the Wiener internuncial pool model for message-passing between neurons in the human nervous system [68] and, more recently, the swarm intelligence model [4].

This chapter is organized as follows. An overview of a granular approach to rough-neurocomputing is presented in Sect. 2. A number of different forms of neurons are briefly described in Sect. 3. The architectures of hybrid forms of neural networks are described in Sect. 4.

2 Granular Approach to Rough-Neurocomputing

A brief introduction to a rough-neurocomputing 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 [44]. A fundamental feature of this model is the design of neurons that engage in knowledge discovery. Mechanically, such neurons return granules (synthesized knowledge) derived from input granules.

2.1 Adaptive Calculus of Granules

To facilitate reasoning about rough neural networks, an *adaptive calculus of granules* has been introduced [40, 44, 48, 49]. The calculus of granules is a system for approximating, combining, describing, measuring, reasoning about, and performing operations on granules by intelligent computing units called agents. In the calculus of granules, the term *information granule* (or *granule*, for short) denotes an assemblage of objects aggregated together by virtue of their indistinguishability, similarity, or functionality. Intuitively, a granule is also called a clump [70]. The term *calculus* comes from G.W. v. Leibniz, who thought of a calculus as an instrument of discovery inasmuch as it provides a system for combining, describing, measuring, reasoning about, and performing operations on objects of interest such as terms in a logical formula in a logical calculus or infinitesimally small quantities in differential calculus [3, 13]. The calculus of classes described by Alfred Tarski [67] shares some of the features found in a calculus of granules. The term *class* is synonymous with *set*, an assemblage of distinct entities, either individually specified or satisfying certain specified conditions [6] (e.g., equivalence class of y consisting of all objects equivalent to y). In measure theory, a class is a set of sets [8]. The element of a class

is a subset. It is Georg Cantor's description of how one constructs a set that comes closest to what we have in mind when we speak of a granulation, that is, a set is the result of collecting together certain well-determined objects of our perception or our thinking into a single whole (the objects are called elements of a set) [7]. In a calculus of classes, the kinds of classes (e.g., empty class, universal class), relations between classes (e.g., inclusion, overlap, identify), and operations on classes ($\cup$, $\cap$, -) are specified. Similarly, a calculus of granules distinguishes among kinds of granules (e.g., elementary granules, set-, concept-, and granule-approximations), relations among granules (e.g., inclusion, overlap, closeness), and operations on granules (e.g., granule approximation, decomposition). It should be observed that in the case of information granules, we cannot use crisp equality in comparing granules. Instead, we are forced to deal with similarity, closeness, and being a part of a whole to a degree, concepts in considering relations between granules.

Calculus of granules includes a number of features not found in the calculus of classes, namely, a system of agents, communication of granules of knowledge between agents, and the construction of granules by agents. To some extent, the new calculus of granules is similar to the agent-based, value-passing calculus of communicating systems proposed by Robin Milner [17, 18]. In Milner's system, an agent is an independent process possessing input and output ports. Agents communicate via channels connecting the output (input) port of one agent with the input (output) port of another agent. Milner's calculus is defined by a tuple $(A, L, Act, X, V, K, J, \varepsilon)$ where A is a set of names; L, a set of labels; Act, a set of actions; X, a set of agent variables; V, a set of values; K, a set of agent constants; J, an indexing set; and ε is a set of agent expressions. This calculus includes a grammar for formulating expressions. Even though adaptivity, granules of knowledge, information granulation, parameterized approximations, and hierarchy of relations of being a part, to a degree (fundamental features of the calculus of granules), are not found in Milner's calculus, it is possible to enrich Milner's system to obtain a variant of the calculus of granules.

The fundamental feature of a granulation system is the exchange of information granules of knowledge between agents by transfer functions induced by rough mereological connectives extracted from information systems. A calculus of granules has been introduced to provide a foundation for the design of information granulation systems. The keystone in such systems is the granularity of knowledge for approximate reasoning by agents [42]. Approximate reasoning on information granules is not only caused by inexactness of information that we have but also by the fact that we can gain efficiency in reasoning if it is enough to deliver approximate solutions, sufficiently close to ideal solutions. An agent is modeled as a computing unit that receives input from its sensors and from other agents, acquires knowledge by discovering (constructing) information granules and by granule approximation, learns (improves its skill in acquiring knowledge), and adapts (adjusts in granulation parameters of predicates in response to changing, for example, sensor measurements

and feedback from other agents). For two finite sets $X, Y \subseteq U$ (universe of an information system), we define standard rough inclusion using

$$\mu(X,Y) = \frac{card\,(X \cap Y)}{card(X)} \text{ if } X \text{ is nonempty, and } \mu(X,Y) = 1, \text{ otherwise.} \quad (1)$$

A simple granule of knowledge of type (μ, B, C, tr, tr') has the form (α, α') where μ is the standard rough inclusion, B and C are subsets of A (attributes, that is, sensors, of an information system), and $tr, tr' \in [0,1]$ are thresholds on functions defined with respect to μ such that $\mu([\alpha]_B, [\alpha']_C) \geq tr$ and $\mu([\alpha']_C, [\alpha]_B) \geq tr'$. For example, we assert that $Gr(\mu, B, C, tr, tr', \alpha, \alpha')$ is true when (α, α') is a (μ, B, C, tr, tr') granule of knowledge. There are several sources of adaptivity in the scheme defined by a calculus of granules. First, there is the possibility that changes can be made in parameters μ, B, C, tr, tr' in the granulation predicate $Gr(\mu, B, C, tr, tr', \alpha, \alpha')$ for any agent $ag \in Ag$ (set of agents). Second, new granules can be constructed by any agent in response to a changing environment. Third, new rough inclusion measures can be instituted by an agent by changing, for example, the parameters in a t-norm and an s-norm used in defining μ. The possibility that any agent can make one or more of these changes paves the way toward an adaptive calculus of granules [42]. A recently formulated rough-fuzzy neural network has partially realized this idea with an adaptive threshold relative to a set of real-value attributes without employing rough inclusion [19].

Each agent (neuron) distills its knowledge from granulated (fused) sensor measurements, from granulated signals from other agents, and from approximate reasoning in classifying its acquired granules. An agent communicates its knowledge through channels connected to other agents. An agent (neuron) learns by adjusting accessible parameters in response to feedback from other agents. Let Ag be a nonempty set of agents. In describing the elements of a calculus of granules, we sometimes write U instead of $U(ag)$, for example, where U [and $U(ag)$] denotes a nonempty set of granules (universe) known to agent $ag \in Ag$. Similarly, when it is clear from the context, we sometimes write Inv, St, A, M, L, $link$, O, AP_O, Unc_rel, Unc_rel, H, Dec_rule, lab as a shorthand for $Inv(ag)$, $St(ag)$, $A(ag)$, $M(ag)$, $L(ag)$, $Link(ag)$, $O(ag)$, $AP\_O(ag)$, $Unc\_rel(ag)$, $Unc\_rule(ag)$, $H(ag)$, $Dec\_rule(ag)$, respectively. The calculus of granules establishes a scheme for a distributed system of agents that is characterized by the following tuple:

$$\begin{aligned} Scheme = (&U, Inv, St, Ag, L_{rm}, A, M, L, link, \\ &O, AP\_O, Unc\_rel, Unc\_rel, H, Dec\_rule, lab), \end{aligned} \quad (2)$$

where U denotes a nonempty set of granules (universe) known to agent $ag \in Ag$, Inv denotes an inventory of elementary objects available to ag, St a set of standard objects for ag, Ag a set of agents, L_{rm} a rough mereological logic [12], A an information system of ag, M a pre-model of L_{rm} for ag, L a set of unary predicates of ag, $link$ a string denoting a team of agents communicating objects (input) to an agent for granulation, O a set of operations of an agent, Unc_rel a set of uncertainty

relations, H a strategy for producing uncertainty rules from uncertainty relations, Dec_rule a set of granule decomposition rules, and lab a set of labels (one for each agent $ag \in Ag$). The calculus of granules provides a computational framework for designing neural networks in the context of a rough set approach to approximate reasoning and knowledge discovery. The original idea of an open-world model for inductive learning by agents [26] has been enriched by considering a distributed system of agents that stimulate each other by communicating granules of knowledge gleaned from granules received from other agents.

An approximate rough mereology with its own logic L_{rm} (syntax, grammar for its formulas, axioms, and semantics of its models) provides a formal treatment of being a part in a degree. This paves the way toward a study of granule inclusion degree testing and measures of the closeness of granules implemented by cooperating agents [44]. The calculus of granules is considered adaptive to the extent that the construction of information granules by a distributed system of interacting agents will vary in response to variations in the approximate reasoning by agents about their input signals (input granules). Agents usually live and learn inductively in an open system like that described by Pawlak [26]. Let (Inv,Ag) denote a distributed system of agents where Inv denotes an inventory of elementary objects and Ag is a set of intelligent computing units (agents). Let $ag \in Ag$ be an agent endowed with tools for reasoning and communicating with other agents about objects within its scope. These tools are defined by components of the agent label (denoted lab) such that

$$\begin{aligned} lab(ag) = [&\mathbf{A}(ag), \mathrm{M}(ag), L(ag), Link(ag), St(ag), O(ag), \\ &AP\_O(ag), Unc\_rel(ag), Unc\_rule(ag), H(ag), Dec\_rule(ag)], \end{aligned} \tag{3}$$

where

- $\mathbf{A}(ag) = [U(ag), A(ag)]$ is an information system relative to agent ag, where the universe $U(ag)$ is a finite, nonempty set of granules containing elements of the form $(\alpha, [\alpha])$ such that α is a conjunction of descriptors and $[\alpha]$ denotes its meaning in $\mathbf{A}(ag)$ [26]. It is also possible that the objects of $U(ag)$ are complex granules.
- $M(ag) = [U(ag), [0,1], \mu_0(ag)]$ is a premodel of L_{rm} with a rough inclusion $\mu_0(ag)$ on the universe $U(ag)$. The notation L_{rm} denotes a rough mereological logic [42].
- $L(ag)$ is a set of unary predicates (properties of objects) in a predicate calculus interpreted in the set $U(ag)$. Further, formulas of $L(ag)$ are constructed as conditional formulas of logics L_B where $B \subset U(ag)$.
- $Link(ag)$ is a collection of strings of the form $ag_1 ag_2 \ldots ag_k ag$ denoting a team of agents such that $ag_1 ag_2 \ldots ag_k$ are the children of agent ag in the sense that ag can assemble complex objects (constructs) from simpler objects sent by agents ag_1, ag_2,...,ag_k [19].
- $St(ag) = \{st(ag)_1, \ldots, st(ag)_n\} \subset U(ag)$ is the set of standard objects at ag.
- $O(ag) \subseteq \{o |\ o : U(ag_1) \times (ag_2) \times \cdots \times U(ag_k) \to U(ag)\ is\ operation\ at\ ag\}$.

- $AP\_O(ag)$ is a collection of pairs of the form

$$\langle o(ag,t), \{AS_1[o(ag),in], \ldots, AS_k[o(ag),in], AS[o(ag),out]\}\rangle,$$

where $o(ag,t) \in O(ag)$, k is the arity of $o(ag)$, $t = ag_1, ag_2, \ldots, ag_k \in Link(ag)$, $AS_i[o(ag,t),in]$ is a parameterized approximation space corresponding to the i-th argument of $o(ag,t)$ and $AS[o(ag,t),out]$ is a parameterized approximation space for the output of $o(ag,t)$. The meaning of $o(ag,t)$ is that an agent performs an operation enabling the agent to assemble from objects $x_1 \in U(ag_1)$, $x_2 \in U(ag_2)$, ..., $x_k \in U(ag_k)$ the object $z \in U(ag)$ that is an approximation defined by $AS[o(ag,t),out]$ of $o(ag,t)(y_1,y_2,\ldots,y_k) \in U(ag)$ where y_i is the approximation of x_i defined by $AS_i[o(ag,t),in]$. One may choose here either a lower or an upper approximation. For more details, refer to Chap. 3.
- $Unc\_rel(ag)$ is a set of uncertainty relations unc_rel_i of type

$$\begin{aligned}[o_i(ag,t), \rho_i(ag), ag_1, \ldots, ag_k, ag,& \\ \mu_o(ag_1), \ldots, \mu_o(ag_k), \mu_o(ag),& \\ st(ag_1)_i, \ldots, st(ag_k)_i, st(ag)_i]&\end{aligned} \tag{4}$$

of agent ag where $ag_1, ag_2, \ldots, ag_k \in Link(ag)$, $o_i(ag,t) \in O(ag)$ and ρ_i is such that $\rho_i[(x_1,\varepsilon_1),\ldots,(x_i,\varepsilon_k),(x,\varepsilon)]$ holds for $x \in U(ag)$, $x_1 \in U(ag_1)$, ..., $x_k \in U(ag_k)$, $\varepsilon,\varepsilon_1,\ldots,\varepsilon_k \in [0,1]$ iff $\mu_o[x_j, st(ag_j)_i] \geqslant \varepsilon_j$, $j = 1,\ldots,k$ for the collection of standards $st(ag_1)_i,\ldots,st(ag_k)_i, st(ag)_i$ such that $o_i(ag,t)[st(ag_1)_i,\ldots, st(ag_k)_i] = st(ag)_i$. Values of the operation o are computed in three stages. First, approximations of input objects are constructed. Next, an operation is performed. Finally, the approximation of the result is constructed. A relation unc_rel_i provides a global description of this process. In practice, unc_rel_i is composed of analogous relations corresponding to the three stages. The relation unc_rel_i depends on parameters of approximation spaces. Hence, to obtain satisfactory decomposition (similarly, uncertainty, and so on) rules, it is necessary to search for satisfactory parameters of approximation spaces. This search is analogous to weight tuning in traditional neural computations.
- $Unc\_rule(ag)$ is a set of uncertainty rules unc_rule_i of type,

$$\begin{aligned}&\textbf{if } o_i(ag,t)[st(ag_1)_i,\ldots,st(ag_k)_i] = st(ag)_i \text{ and} \\ &\quad x_1 \in U(ag_1),\ldots,x_k \in U(ag_k) \text{ satisfy the conditions} \\ &\quad \mu_o[x_j, st(ag_j)_i] \geqslant \varepsilon(ag_i) \text{ for } i = 1,\ldots,k, \\ &\textbf{then } \mu_o[o_i(ag,t)(x_1,\ldots,x_k), st(ag)_i] \geqslant f_i[\varepsilon(ag_1),\ldots,\varepsilon(ag_k)],\end{aligned} \tag{5}$$

where $ag_1, ag_2, \ldots, ag_k \in Link(ag)$ and $f_i : [0,1]^k \to [0,1]$ is a so-called rough mereological connective. Uncertainty rules provide functional operators (approximate mereological connectives) for propagating uncertainty measure values from the children of an agent to the agent. The application of uncertainty rules is in negotiation processes where they inform agents about plausible uncertainty bounds.

- $H(ag)$ is a strategy that produces uncertainty rules from uncertainty relations.
- $Dec\_rule(ag)$ is a set of decomposition rules,

$$[\Phi(ag_1), ..., \Phi(ag_k), \Phi(ag)], \tag{6}$$

of type $[o_i(ag,t), ag_1, ..., ag_{k,}ag]$ of agent ag, where

$$\Phi(ag_1) \in L(ag_1), ..., \Phi(ag_k) \in L(ag_k), \Phi(ag) \in L(ag), \tag{7}$$

$ag_1, ag_2, \ldots, ag_k \in Link(ag)$, and there exists a collection of standards $st(ag_1)_i$, $\ldots$, $st(ag_k)_i, st(ag)_i$ such that $o_i(ag,t)[st(ag_1)_i, \ldots, st(ag_k)_i] = st(ag)_i$ and these standards satisfy $\Phi(ag_1), \ldots, \Phi(ag_k), \Phi(ag)$, respectively. Decomposition rules are decomposition schemes, that is, such rules describe the standard $st(ag)_i$ and standards $st(ag_1)_i, \ldots, st(ag_k)_i$ from which the standard $st(ag)_i$ is assembled under o_i relative to predicates that these standards satisfy.

It has been pointed out that there is an analogy between calculi of granules in distributed systems and rough-neurocomputing [44]:

1. An agent *ag* with input and output ports creating 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 [17].
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.

2.2 Granules in Distributed Systems

In this section, the fulfillment of an ontology of approximate reasoning stems from the consideration of granular computing in the context of parameterized approximation spaces as a realization of an adaptive granule calculus. This realization is made possible by introducing a parameterized approximation space in designing a reasoning system for an agent. A step toward the realization of an adaptive granule calculus in a rough-neurocomputing scheme is described in this section and is based on [44]. In a scheme for information 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.

Let us now consider a definition of a parameterized approximation space. A parameterized approximation space is a system

$$AS_{\#,*,\$} = (U, I_\#, R_*, \nu_\$), \tag{8}$$

where #,*, $ denote vectors of parameters, U is a nonempty set of objects, and

- $I_\# : U \to \wp(U)$ is an *uncertainty function* where $\wp(U)$ denotes the power set of U; $I_\#(x)$ is called the *neighborhood* of $x \in U$;
- $R_* \subseteq \wp(U)$ is a family of *parameterized patterns*;
- $\nu_\$: \wp(U) \times \wp(U) \to [0,1]$ denotes *rough inclusion*.

The uncertainty function defines for every object x in U a set of similarly described objects. 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 (parameterized) patterns (e.g., representing, for fixed values of parameters, 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 patterns. The rough inclusion function $\nu_\$$ defines the value of inclusion between two subsets of U. In particular, for any neighborhood, its inclusion degree in a given pattern can be computed. Moreover, for classifiers, the degree of inclusion of patterns in decision classes can be estimated. The neighborhood $I_\#(x)$ can usually be defined as a collection of objects close to x. 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.

For a parameterized approximation space $AS_{\#,*,\$}$ and any subset $X \subseteq U$, the lower and upper approximations of X in U based only on an uncertainty function and rough inclusion are defined as follows:

$$LOW\left(AS_{\#,*,\$}, X\right) = \{x \in U |\ \nu_\$\left(I_\#(x), X\right) = 1\}\ [\textit{lower approximation}], \tag{9}$$

$$UPP\left(AS_{\#,*,\$}, X\right) = \{x \in U |\ \nu_\$\left(I_\#(x), X\right) > 0\}\ [\textit{upper approximation}]. \tag{10}$$

However, if one would like to consider the approximation of concepts in an extension U' of U by taking patterns and their inclusion degrees in the concepts, the definition of concept approximation should be changed. The reader can find more details on concept approximations in Chaps. 3, 6, and 25.

Sets of objects that are collections of objects defined by an uncertainty function or patterns from a data table are examples of information granules. A parameterized approximation space can be treated as an analogy to a neural network weight (see Fig. 1). In Fig. 1, $w_1, \ldots, w_n$, Σ, f denote the weights, aggregation operator, and activation function of a classical neuron, respectively, whereas $AS_1(P), \ldots, AS_k(P)$ denote parameterized approximations spaces where agents process input granules $G_1, \ldots, G_k$ and O denotes a (parameterized) operation from a given set of operations that produce the output of a granular network. The parameters in P of an approximation space should be learned to induce the relevant information granules.

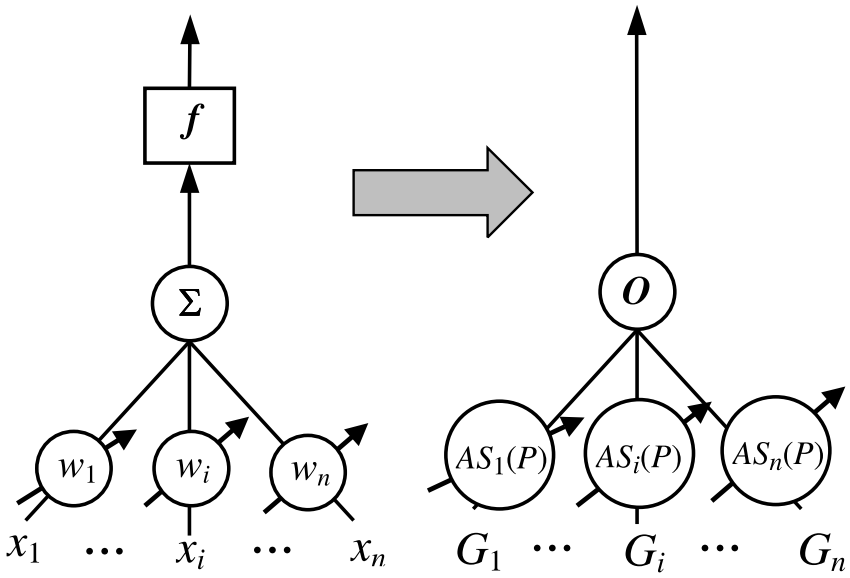

Fig. 1. Comparison of classical and granular network architectures

3 Rough Neurons

The term *rough neuron* was introduced in 1996 by Lingras [15]. In its original form, a rough neuron was defined relative to upper and lower bounds, and inputs were assessed relative to boundary values. Hence, this form of neuron might also be called a boundary value neuron. This form of rough neuron has been used in predicting urban high-traffic volumes [4]. More recent work considers rough-neural networks (RNNs) with neurons that construct rough sets and output the degree of accuracy of an approximation [35, 36]. This has led to the introduction of approximation neurons [36] and their application in classifying electrical power system faults [10], signal analysis [37], and in assessing software quality [29]. An information granulation model of a rough neuron was introduced by Skowron and Stepaniuk in the late 1990s. This model of a rough neuron is inspired by the notion of a cooperating agent (neuron) that constructs granules; perceives by measuring values of available attributes, granule inclusion, granule closeness, and by granule approximation; learns by adjusting parameters in its local parameter space; and shares its knowledge with other agents (neurons). A rough-fuzzy multilayer perceptron (MLP) useful in knowledge encoding and classification was introduced in 1998 by Banerjee, Mitra, and Pal [2]. The study of various forms of rough neurons is part of a growing number of papers on neural networks based on rough sets. Transducers discussed in Chap. 8 transforming rough set arguments into rough sets can also be considered as rough neurons.

3.1 Set Approximation

Rough set theory offers a systematic approach to set approximation [26]. 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 $Ind_A(B)$ such that

$$Ind_A(B) = \{(x,x') \in U^2 \,|\, \forall a \in B.a(x) = a(x')\}. \tag{11}$$

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 symbol $[x]_B$ denotes the equivalence class of $\mathrm{Ind}_A(B)$ defined by x. Further, partition symbol $\mathrm{U/Ind}_A(B)$ 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 B-upper approximation denoted by $\underline{B}X$ and $\overline{B}X$, respectively, where

$$\underline{B}X = \{x|[x]_B \subseteq X\} \text{ and } \overline{B}X = \{x|[x]_B \cap X \neq \emptyset\}. \tag{12}$$

3.2 Rough Membership Set Function

In this section, a set function form of the traditional rough membership function introduced in [54] is applied. 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 $\mathrm{Ind}_A(B)$. A set function $\mu_u^B : \wp(U) \to [0, 1]$ defined by (13)

$$\mu_u^B(X) = \frac{card(X \cap [u]_B)}{card([u]_B)} \tag{13}$$

for any $X \in \wp(Y)$, $u \in U$, is called a *rough membership function*.

A rough membership function provides a classification measure inasmuch as it tests the degree of overlap between the set X and the equivalence class $[u]_B$. The form of rough membership function presented above is slightly different from the classical definition [27], where the argument of the rough membership function is an object u and the set X is fixed. For example, let $X_{B_{\mathrm{approx}}} \in \{\overline{B}X, \underline{B}X\}$ denote a set approximation. Then, we compute the degree of overlap between $X_{B_{\mathrm{approx}}}$ and $[u]_B$ by

$$\mu_u^B(X_{B_{\mathrm{approx}}}) = \frac{card([u]_B \cap X_{B_{\mathrm{approx}}})}{card([u]_B)}. \tag{14}$$

In the sequel, we also write $\mu_{u,B}(X_{B_{\mathrm{approx}}})$ instead of $\mu_u^B(X_{B_{\mathrm{approx}}})$.

3.3 Decision Rules

In deriving decision system rules, the discernibility matrix and discernibility function are essential. Given an information system $S = (U,A)$ with n objects, the $n \times n$ matrix (c_{ij}), called the discernibility matrix of S [denoted $M(S)$], is defined as

$$c_{ij} = \{a \in A |\ a(x_i) \neq a(x_j)\},\ \text{for } i,j = 1,\ldots,n. \tag{15}$$

A discernibility function $f_{M(S)}$ for the 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 by

$$f_{M(S)}(a_1^*,\ldots,a_m^*) = \wedge\{c_{ij}^* |\ 1 \leq j < i \leq n, c_{ij} \neq \emptyset\} \text{ where } c_{ij}^* = \{a^* |\ a \in c_{ij}\}. \tag{16}$$

Precise conditions for decision rules can be extracted from a discernibility matrix as in [43, 47]. For the information system $S = (U,A)$, 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 $\delta \in A - B$, a decision function $d_\delta^B : U \to \wp(V_\delta)$ is defined in (17) as in [56]:

$$d_\delta^B(u) = \left\{v \in V_\delta |\ \exists u' \in U, \left(u',u\right) \in Ind_B \text{ and } \delta(u') = v\right\}. \tag{17}$$

In other words, $d_\delta^B(u)$ is the set of all elements of the decision column δ 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 following form:

$$a_{i_1}(u_i) = v_{i_1} \wedge \ldots \wedge a_{i_j}(u_i) = v_{i_j} \wedge \ldots \wedge a_{i_r}(u_i) = v_{i_r} \underset{S}{\Rightarrow} d(u_i) = v, \tag{18}$$

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

$$(a_{i_1} = v_{i_1}) \wedge \ldots \wedge (a_{i_r} = v_{i_r}) \underset{S}{\Rightarrow} (a_p = v_p). \tag{19}$$

In practice also are important rules that are true in S to the degree in which the set defined in S by the left-hand side of the rule is included in the set defined in S by the right-hand side of the rule. The left- and right-hand sides of rules are information granules in S. Then the degree mentioned above can be interpreted as the degree of inclusion of such information granules. The decision rules can also be treated as information granules (see Chap. 3).

Let $RED(S)$ be a reduct set generated from a decision system S, e.g., a set of local reducts with respect to objects [12]. [1] For decision system S, the set of decision

[1] Note that there are many different kinds of reducts and methods of selection of relevant reducts used in constructing data description models (see Chap. 25).

rules constructed with respect to a reduct $R \in RED(S)$ is denoted by $OPT(S,R)$. Then the set $OPT(S)$ of all decision rules derivable from reducts in $RED(S)$ is the following set:

$$OPT(S) = \cup\{OPT(S,R) |\ R \in RED(S)\}. \tag{20}$$

3.4 Interval-Based Rough Neuron

An *interval-based rough neuron* was introduced in 1996 [15]. A brief introduction to this form of rough neuron is given in this section. Rough neurons are defined in the context of rough patterns. Objects such as a fault signal or daily weather can be described by a finite set of features (e.g., amplitude, type of waveform, high-frequency component, rainfall, temperature) characterizing each object. The description of an object is an n-dimensional vector, where n is the number of features used to characterize an object. A pattern is a class of objects based on the values of some features of objects belonging to the class.

Let x be a feature variable in the description of an object. Further, let $\underline{x}, \overline{x}$ represent upper and lower bounds of x. In a rough pattern, the value of each feature variable x is specified by $\underline{x}, \overline{x}$ (called rough values). Rough values are useful in representing an interval or set of values for a feature, where only the upper and lower bounds are considered relevant in a computation. This form of rough neuron can be used to process intervals in a neural network.

Let $r, \underline{r}, \overline{r}$ denote a rough neuron, lower neuron, and upper neuron, respectively. A rough neuron is a pair $(\underline{r}, \overline{r})$ with three types of connections: i/o connections to $\underline{r}$, i/o connections to $\overline{r}$, and connections between $\underline{r}$ and $\overline{r}$. In effect, a rough neuron stores the upper and lower bounds of input values for a feature and uses these bounds in its computations. Let in_i, out_j, w_{ij} denote the input to neuron i, the output from neuron j, and the strength of the connection between neurons i and j, respectively. The input to an upper, lower, or conventional neuron i is calculated as a weighted sum as

$$in_i = \sum_{j=1}^{n} w_{ij} out_j \quad (\text{neuron} j \text{ is connected to neuron } i). \tag{21}$$

Assuming the subscript $i = \underline{r}$, we obtain the input to a lower neuron, and for $i = \overline{r}$, we obtain the input to an upper neuron. Let t be a transfer function used to evaluate the input to an upper (lower) neuron. Then the output of an upper (lower) neuron is computed as in (22) and (23), respectively:

$$out_{\overline{r}} = \max\left[t\left(in_{\overline{r}}\right), t\left(in_{\underline{r}}\right)\right]; \tag{22}$$

$$out_{\underline{r}} = \min\left[t\left(in_{\overline{r}}\right), t\left(in_{\underline{r}}\right)\right]. \tag{23}$$

The output of a rough neuron will be computed from

$$rough\_neuron\_output = \frac{out_{\overline{r}} - out_{\underline{r}}}{average(out_{\overline{r}}, out_{\underline{r}})}. \tag{24}$$

The inputs to rough neurons considered in [15] are related to deviations in measurements of some attribute value. One can consider another case when deviations of a real function defined, e.g., on the lower approximation of a given set X, are used to define inputs to neurons. Another possibility to consider is deviations of rough membership function values on elements of a tolerance class (see Sect. 3.5).

3.5 Approximation Neurons

This section considers the design of rough neural networks based on set approximations and rough membership functions, and hence, this form of network is called an *approximation neuron* (AN). The approximation neuron was introduced in [9], and elaborated in [35, 36]. Preliminary computations in an AN are carried out with a layer of approximation neurons, which construct rough sets and where the output of each approximation neuron is computed with a rough membership function. This section considers ANs constructed with one type of rough neuron: the approximation neuron. Let $B, F, F_{B_{\text{approx}}}$, $[f]_B$ denote a set of attributes, a finite set of neuron inputs (this is an archival set representing past stimuli, a form of memory accessible to a neuron), a set approximation, and an equivalence class containing measurements derived from known objects, respectively. The basic computation steps performed by an approximation neuron are illustrated in Fig. 2.

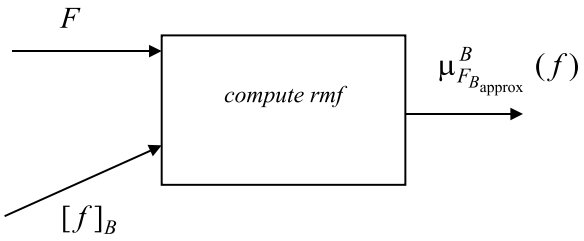

Fig. 2. Approximation neuron

The approximation neuron measures the degree of overlap of a set $[f]_B$ and $F_{B_{\text{approx}}}$. Let us consider a more general case when instead of an indiscernibility class $[f]_B$, an input is defined by a more general information granule (see Chap. 3), i.e., τ-tolerance class of $[f]_B$, (a family $\{[f']_B : f \tau f'\}$). The output of a neuron is defined by two numbers representing the deviation of the rough membership function on elements of the tolerance class. Other forms of rough neurons are described in [37].

3.6 Decider Neuron

The notion of a *decider neuron* was introduced in [35, 36] and applied in [37]. A decider neuron implements a collection of decision rules by (i) constructing a

condition vector c_{exp} from its inputs, which are rough membership function values, (ii) discovering the rule $c_i \Longrightarrow d_i$ with a condition vector c_i that most closely matches an input condition vector c_{exp}, and (iii) outputs $AND(1-e_i, d_i)$ where $d_i \in \{0,1\}$, and relative error $e_i = \|c_{\text{exp}} - c_i\|/\|c_i\| \in [0,1]$ where $\|\cdot\|$ denotes the vector length function. When $e_i = 0$, then $y_{\text{rule}} = AND(1-e_i, d_i) = d_i$, and the classification is successful. If $e_i = 1$, then $y_{\text{rule}} = AND(1-e_i, d_i) = 0$ indicates the relative error in an unsuccessful classification. A flow graph showing the basic computations performed by a decider neuron is given in Fig. 3.

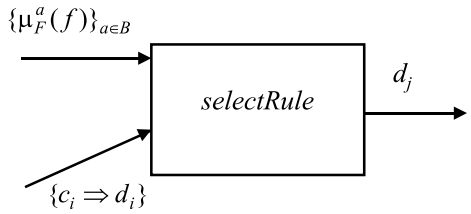

Fig. 3. Flow graph for decider neuron

The set $rmf = \{\mu_F^a(f)\}_{a\in B}$ consists of approximation neuron measurements in response to the stimulus provided a new object f requiring classification. The elements of the set rmf are used by a decider neuron to construct an experimental condition vector c_{exp}. A second input to a decider neuron is the set $R = \{c_i \Longrightarrow d_i\}$. The elements of the set R are rules that have been derived from a decision table using rough set theory. Let *selectRule* denote a process that implements an algorithm to identify a condition vector in one of the rules of R, that most closely matches c_{exp}.

Let us observe that one of the input of decider neuron is an information granule represented by a set of decision rules. The rough neuron considered is used to measure the degree of closeness of object f to an information granule represented by the set of decision rules.

It is worthwhile mentioning that the case considered is very simple. Instead of a rough membership function, computed relative to attributes from B, one should consider a relevant family $C_1, \ldots, C_k$ of subsets of B. For classifiers based on decision rules, such subsets are defined by reducts local with respect to objects [12]. [2]

3.7 Architecture of Approximation Neural Networks

Approximation neural networks (ANN) are well suited to solving classification problems where nuances in a feature space over time can be gleaned from rough approximations. This form of rough neural computation has been successfully applied

[2] In the more general case, one can consider, instead of *rmf* inclusion, degrees of input information granules into the information granules representing relevant patterns for decision classes (see Chaps. 3 and 25).

in two applications: classifying the waveforms of electrical power system faults ([9], [10, 37] and in determining the number of changes required in a software system based on quality measurements [31]. In this section, a brief summary of the results of the software quality study are given.

Sample computations with a set of 11 approximation neurons are given in Table 1. The first 11 columns of Table 1 are rough membership function outputs, and the last column is a target value for the aggregate of the rmf values. The target column in Table 1 serves as a decision column.

Table 1. Sample approximation neuron output values

$\mu_u^{a_1}(X)$	$\mu_u^{a_2}(X)$	$\mu_u^{a_3}(X)$	$\mu_u^{a_4}(X)$	$\mu_u^{a_5}(X)$	$\mu_u^{a_6}(X)$	$\mu_u^{a_7}(X)$	$\mu_u^{a_8}(X)$	$\mu_u^{a_9}(X)$	$\mu_u^{a_{10}}(X)$	$\mu_u^{a_{11}}(X)$	**Target**
1	1	1	1	1	0.9	0.9	0.72	0.47	0.66	1	**1**
1	1	1	1	1	0.9	0.9	0.72	0.4	0.66	1	**1**
						...					
1	1	1	1	1	0.9	0.927	0.72	0.47	0.66	1	**1**
0.68	0.89	0.74	0.78	0	0	0	0	0.57	0	0.23	**0**

During calibration of the ANN, adjustments are made to the weights (strengths of connections) associated with approximation neurons relative to a probabilistic sum of the rmf values and target value. For each test input, the output of an ANN indicates the probability that the test input belongs to the set of measurements associated with the *i*th life cycle product attribute. Let $\sum$ denote an output neuron (see Fig. 4) that computes the sum of all weighted outputs from the approximation neurons in the first layer. A software metric (e.g., reusability, complexity, maintainability [6]) is used to define each life cycle product quality attribute.

The application of the rough mereological approach in software quality measurement is described in [38]. The basic idea is to characterize the global quality assessment of a given software product using a vector of rough mereological distances (cf. rows of Table 1) of quality measurements relative to a chosen set of industry standards. Sample calibrations of a rough membership function neural network are shown in Figs. 5 and 6. In the case considered, constructed information granules are degrees of inclusion of patterns in decision classes. They are computed as linear combinations of rough membership degrees of a given input u in the set X.

3.8 Architecture of Approximation-Decider Neural Network

The output of the approximation neural network in Fig. 4 serves as a module in testing the quality measurements of a particular life cycle product relative to a particu-

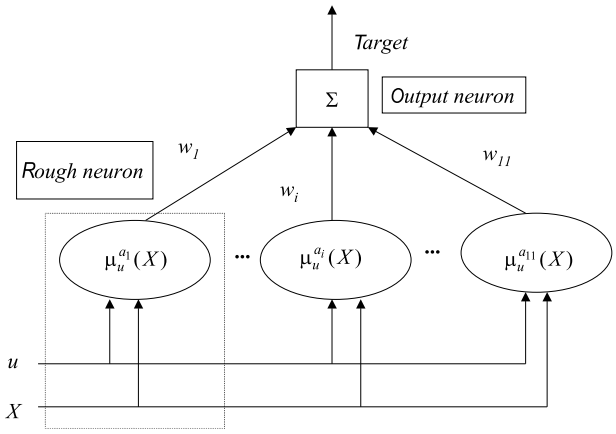

Fig. 4. Approximation neural network

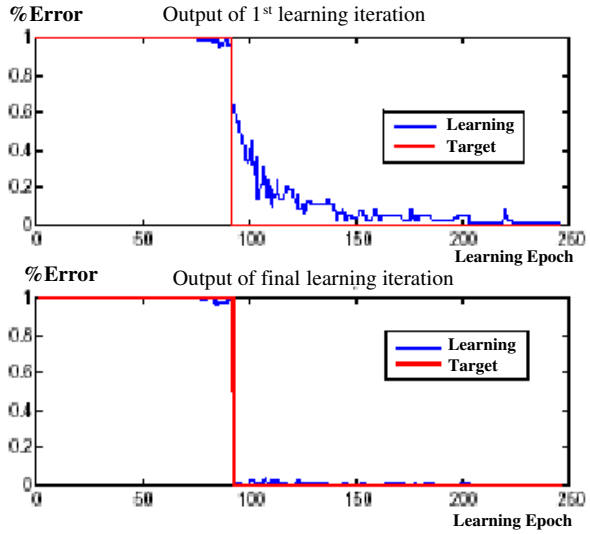

Fig. 5. Sample calibration of approximation neural network

lar number of changes required to correct product quality deficiencies. For the study of the quality of life cycle products, 17 such approximation neural networks were constructed. The outputs of the k (k = 17) ANNs form a condition vector of the form $[e_1\ e_2 \ldots e_k]$, where e_i denotes an experimental approximation neuron output (rough membership function) value. During training, a test set of condition vectors and corresponding decisions (specified number of changes decided on for each condition vector) was constructed. A set of decision rules is then derived and incorporated in a neural network output neuron that is called a decider neuron (see Fig. 7). During

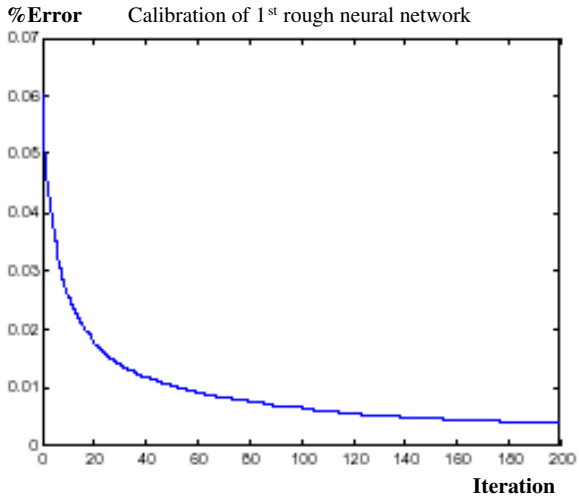

Fig. 6. Sample calibration of approximation neural network (cd.)

testing, each test condition vector $[t_1 \ldots t_k]$ is matched with the closest training set condition vector in the set of rules in the decider neuron. The output of the composite rough neural network is the decision of a selected rule in the decider neuron (see Fig. 8). The design of a neural network with a layer of rule-based neurons has been used in a feed-forward multilayer neural network [64].

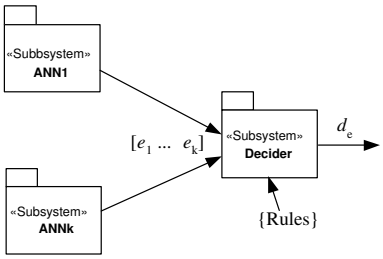

Fig. 7. Training network

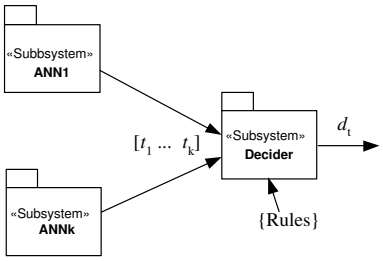

Fig. 8. Testing network

For the experiment described in this section, 17 approximation neural network modules (a total of 187 approximation neurons with a structure like that in Fig. 4) were incorporated into the design of a composite neural network with a single neuron in the output layer, namely, a decider neuron. After calibration of the subnetworks ($ANN_1, \ldots, ANN_k$), formation of a decision table, and derivation of decision rules, the result is the training network in Fig. 7. During testing, the network in Fig. 8 is

used to find a rule with a condition vector with the best match to a rule residing in the decider neuron. As a result, we obtain a basis for making an approximate decision about the number of changes required for the particular life cycle product being evaluated. The results of sample training sessions are shown in Fig. 9.

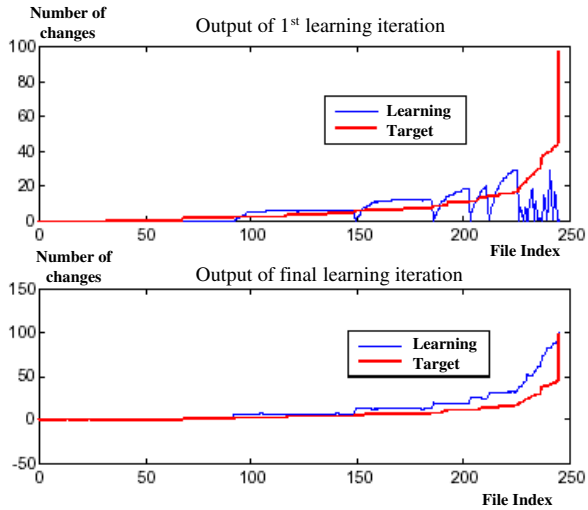

Fig. 9. RNN performance

4 Hybrid Neural Networks

A number of hybrid neural networks with architectural designs based on rough set theory and more traditional neural structures have been proposed: *rough-fuzzy MLP* [19], *evolutionary rough-fuzzy MLP* [24], *interval-based rough-fuzzy networks* [15], and *approximation rough-fuzzy networks* [37]. It should also be mentioned that it is common to use rough set theory as the basis for preprocessing inputs to a neural network (see Chap. 25). In this section, two recent forms of hybrid networks are briefly described: a rough-fuzzy multilayer perceptron neural network [19] and a rough-fuzzy approximation neural network [29]. In rough-fuzzy neural networks, some patterns are extracted using rough set methods. Next, such patterns can be fused using fuzzy logic rather than classical propositional connectives.

4.1 Rough-Fuzzy MLP

The rough-fuzzy multilayer perceptron neural network introduced in [19], was developed for pattern classification. This form of MLP combines both rough sets and

fuzzy sets with neural networks for building an efficient connectionist system. In this hybridization, fuzzy sets help in handling linguistic input information and ambiguity in output decision, whereas rough sets extract domain knowledge for determining network parameters.

4.2 Architecture of Rough-Fuzzy MLP Network

The first step in designing a rough-fuzzy MLP is to establish a basis for working with real-valued attribute tables of fuzzy membership values. The traditional model of a discernibility matrix given in (15) is replaced by

$$c_{ij} = \left\{ a \in B \middle| \left| a(x_i) - a(x_j) \right| > Th \right\} \tag{25}$$

for $i, j = 1, \ldots, n_k$, where Th is an adaptive threshold. Let a_1, a_2 correspond to two membership functions (attributes) where a_2 is steeper compared to a_1 (see Fig. 10). It is observed that $r_1 > r_2$. This results in an implicit adaptivity of Th while computing c_{ij} in the discernibility matrix directly from the real-valued attributes. Herein lies the novelty of the proposed method. Moreover, this type of thresholding also enables the discernibility matrix to contain all representative points/clusters present in a class. This is particularly useful in modeling multimodal class distributions. Rough set methods are used to find patterns relevant to decision classes. While designing

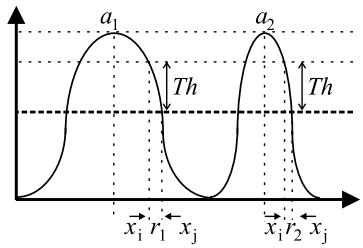

Fig. 10. Illustration of adaptive thresholding of membership functions

the initial structure of the rough-fuzzy MLP, the union of the rules of l classes is considered. The input layer consists of $3n$ attribute values (it is assumed that each attribute can have three fuzzy values: *low, medium, high*) whereas the output layer is represented by l classes. The hidden layer nodes model the first level (innermost) operator in the antecedent part of a rule, which can be either a conjunct or a disjunct. The output layer nodes model the outer level operands, which can again be either a conjunct or a disjunct. For each inner level operator, corresponding to one output class (one dependency rule), one hidden node is dedicated. Only those input attributes that appear in this conjunct/disjunct are connected to the appropriate hidden node, which in turn is connected to the corresponding output node. Each outer level operator is modeled at the output layer by joining the corresponding hidden nodes. Note that a single attribute (involving no inner level operators) is directly connected to the appropriate output node via a hidden node, to maintain uniformity in rule mapping.

Modular Training A method of learning the parameters of rough-fuzzy MLP has recently been described [24] using the modular concept that is based on the divide and conquer strategy. This provides accelerated training and a compact network suitable for generating a minimum number of classification rules with high certainty values. A new concept of a variable genetic mutation operator is introduced for preserving the localized structure of the constitutive knowledge-based subnetworks while they are integrated and evolved.

4.3 Rough-Fuzzy Approximation Neural Network

A rough-fuzzy approximation neural network gains its name from the fact that it contains neurons designed using rough set theory connected to various forms of neurons designed using fuzzy set theory (see, e.g., Fig. 11).

4.4 Architecture of a Sample Rough-Fuzzy Neural Network

The sample rough-fuzzy neural network described in this section consists of four layers (see Fig. 11). The first layer of the network in Fig. 11 contains approximation neurons connected to very basic fuzzy neurons (layer 2) that compute the degree of membership of rough neuron outputs in various distributions. Layer 2 neurons are connected to AND neurons (also called logic neurons [29]). Let x be an approxima-

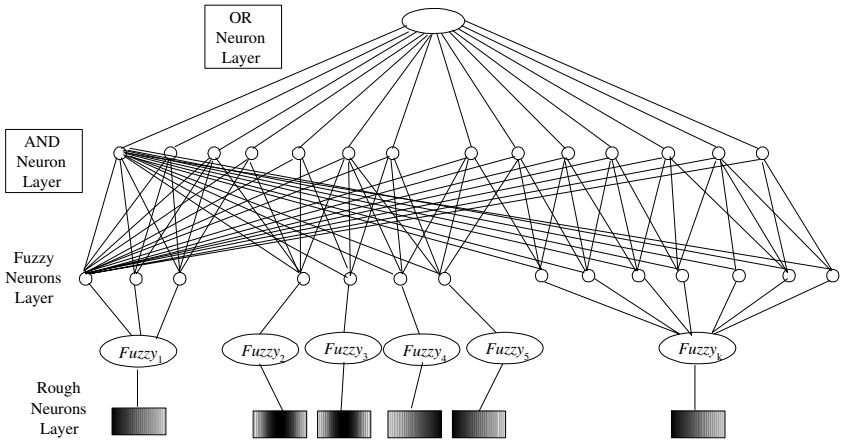

Fig. 11. Rough-fuzzy neural network

tion neuron output, and let $f(x)$ be the degree of membership of x in a particular distribution. Then defined parameters r and w denote a cutoff (reference point) and strength of connection (weight), respectively. Then, we define a fuzzy implication $r \to x$ by

$$(r \to x) = \begin{cases} \frac{x}{r} & \text{if } \ x < r \\ \min\left(1, \frac{x}{r}\right) & \text{otherwise.} \end{cases}$$

Let t ("and" usually interpreted as *min*) and s ("or" interpreted as a probabilistic *sum* here) denote t-norm and s-norm operators from fuzzy set theory [29]. An AND neuron has output z defined as

$$z = \underset{i=1}{\overset{n}{T}} [x_i \, s \, w_i] = \min [x_1 \, s \, w_1, \ldots, x_n \, s \, w_n] .$$

The model for an AND neuron is specialized relative to fuzzy implication as

$$z = \min_{i=1}^{n} \{ [r_i \to f(x)_i] + w_{\mathrm{i}} - [r_i \to f(x)_i] \, w_i \} .$$

The output layer of the network in Fig. 11 consists of OR neurons that aggregate the information gleaned from connections to AND neurons. The model for an OR neuron is

$$y = \underset{i=1}{\overset{n}{S}} [z_i \, \mathrm{t} \, \mathrm{u}_i] = (z_1 \, t \, u_1) \, s \ldots s \, (z_n \, t \, u_{\mathrm{n}}) .$$

In the model for an OR neuron, u_i denotes the strength of connection between the OR neuron and the ith AND neuron.

4.5 Supervised Learning Approach to Calibration of Rough-Fuzzy Neural Network

The calibration scheme for rough-fuzzy neuron networks described in this section employs the standard method for supervised learning. What follows is a brief summary of the calibration steps:

1. Initialize cutoff r and network strength of connections w and u.
2. Introduce a training set.
3. Compute y of the output OR neuron.
4. Compute error Q by comparing network outputs with a target value using (26).

$$Q = target - y. \tag{26}$$

5. Let $\alpha > 0$ denote the positive learning rate. Based on the value error Q in (26), adjust the $r, w,$ and u parameters using the usual gradient-based optimization method suggested in (27) and (28):

$$param(new) = param - \alpha \frac{\partial Q}{\partial param}, \tag{27}$$

$$\frac{\partial Q}{\partial param} = \frac{\partial Q}{\partial y} \frac{\partial y}{\partial param} . \tag{28}$$

A more detailed explanation of how one trains a network containing combinations of AND and OR nodes is given in [31].

5 Concluding Remarks

A scheme for designing rough neural networks based on an adaptive calculus of granules for distributed systems of cooperating agents has been presented. This scheme is defined in the context of an approximate rough mereology, granule construction and granule approximation algorithms, measures of granule inclusion and closeness, and local parameterized approximation spaces. Adaptivity is also a feature of this scheme, where agents can change local parameters in response to changing signals from other agents and from the environment. A number of models of rough neurons have been proposed. Four such models have been briefly described in this chapter: interval-based neurons, approximation neurons, decider neurons, and rough-fuzzy MLPs. Four rough-neurocomputing architectures have also been briefly considered: approximation neural network, approximation-decider neural network, rough-fuzzy neural network, and rough-fuzzy MLP network.

Acknowledgments

The research of James Peters has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) research grant 185986. Lech Polkowski was supported by a grant from the State Committee for Scientific Research of the Republic of Poland (KBN), No. 8T11C02417. The research of Andrzej Skowron has been supported by the State Committee for Scientific Research of the Republic of Poland (KBN), research grant No. 8T11C02519 and by a Wallenberg Foundation grant.

References

1. M.A. Arbib. The artificial neuron. In E. Fiesler, R. Beale, editors, *Handbook of Neural Computation*, B1.1–B1.7, Institute of Physics Publishing, Bristol, 1997.
2. M. Banerjee, S. Mitra, S.K. Pal. Rough fuzzy MLP: Knowledge encoding and classification. *IEEE Transactions on Neural Networks*, 9(6): 1203–1216, 1998.
3. I.M. Bocheński. *A History of Formal Logic*. Chelsea, New York, 1956.
4. E. Bonabeau, M. Dongo, G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Oxford, 2000.
5. B. Chakraborty. Feature subset selection by neuro-rough hybridization. In *[74]*, 519–572, 2001.
6. N.E. Fenton, S.L. Pfleeger. *Software Metrics: A Rigorous & Practical Approach*. PWS, Boston, 1997.
7. W. Gellert, H. Kustner, M. Hellwich, H. Kastner. *The VNR Concise Encyclopedia of Mathematics*. Van Nostrand, London, 1975.
8. P.R. Halmos. *Measure Theory*. Van Nostrand, London, 1950.
9. L. Han, R. Menzies, J.F. Peters, L. Crowe. High voltage power fault–detection and analysis system: Design and implementation. In *Proceedings of the Canadian Conference on Electrical & Computer Engineering (CCECE'99)*, 1253–1258, Edmonton, 1999.
10. L. Han, J.F. Peters, S. Ramanna, R. Zhai. Classifying faults in high voltage power systems: A rough-fuzzy neural computational approach. In *[73]*, 47–54, 1999.

11. T. Kohonen. The self-organizing map. In: *Proceedings IEEE*, 78: 1464–1480, 1990.
12. J. Komorowski, Z. Pawlak, L. Polkowski, A. Skowron. Rough sets: A tutorial. In *[25]*, 3–98, 1999.
13. G.W. Leibniz. In L. Couturat, editor, *Opuscles et Fragments Inedits de Leibniz*, 256, Félix Alcan, Paris, 1903.
14. P.J. Lingras. Fuzzy-rough and rough-fuzzy serial combinations in neurocomputing. *Neurocomputing*, 36: 29-44, 2001.
15. P.J. Lingras. Rough neural networks. In *Proceedings of the 6th International Conference on Information Processing and Management of Uncertainty (IPMU'96)*, 1445–1450, Universidad da Granada, Granada, 1996.
16. P.J. Lingras. Comparison of neofuzzy and rough neural networks. *Information Sciences. An International Journal*, 110: 207–215, 1998.
17. R. Milner. *Communication and Concurrency*. Prentice-Hall, Upper Saddle River, NJ, 1989.
18. R. Milner. *Calculus of Communicating Systems*. Report number ECS-LFCS-86-7 of Computer Science Department, University of Edinburgh, 1986.
19. 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.
20. H.S. Nguyen, A. Skowron, M.S. Szczuka. Situation identification by unmanned aerial vehicle. In *[74]*, 49–56, 2001.
21. H.S. Nguyen, M. Szczuka, D. Ślęzak. Neural networks design: Rough set approach to real-valued data. In *Proceedings of the 1st European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'97)*, LNAI 1263, 359–366, Springer, Berlin, 1997.
22. T. Nguyen, R.W. Swiniarski, A. Skowron, J. Bazan, K. Thagarajan. Applications of rough sets, neural networks and maximum likelihood for texture classification based on singular decomposition. In *Proceedings of the 3rd International Workshop on Rough Sets and Soft Computing (RSSC'94)*, 332–339, San Jose, CA, 1994.
23. S.K. Pal, S. Mitra. Multi-layer perceptron, fuzzy sets and classification. *IEEE Transactions on Neural Networks*, 3: 683–697, 1992.
24. S.K. Pal, P. Mitra. Rough Fuzzy MLP: Modular evolution, rule generation and evaluation. *IEEE Transactions on Knowledge and Data Engineering* (in press).
25. S.K. Pal, A. Skowron, editors. *Rough-Fuzzy Hybridization: A New Trend in Decision Making*. Springer, Singapore, 1999.
26. Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer, Dordrecht, 1991.
27. 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.
28. W. Pedrycz, F. Gomide. *An Introduction to Fuzzy Sets: Analysis and Design*. MIT Press, Cambridge, MA, 1998.
29. 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.
30. Z. Pawlak, J.F. Peters, A. Skowron, Z. Suraj, S. Ramanna, M. Borkowski. Rough measures and integrals: A brief introduction. In *[72]*, 375–379, 2001.
31. W. Pedrycz, J.F. Peters. Learning in fuzzy Petri nets. In J. Cardoso, H. Scarpelli, editors. *Fuzziness in Petri Nets*, 858–886, Physica, Heidelberg, 1998.

32. J.F. Peters, A. Skowron, J. Stepaniuk. Rough granules in spatial reasoning. In *Proceedings of the Joint 9th International Fuzzy Systems Association (IFSA) World Congress and 20th North American Fuzzy Information Processing Society (NAFIPS) International Conference*, 1355–1361, Vancouver, BC, 2001.
33. J.F. Peters, S. Ramanna. A rough set approach to assessing software quality: Concepts and rough Petri net model. In *[25]*, 349–380, 1999.
34. J.F. Peters, W. Pedrycz. *Software Engineering: An Engineering Approach*. Wiley, New York, 2000.
35. J.F. Peters, A. Skowron, Z. Suraj, L. Han, S. Ramanna. Design of rough neurons: Rough set foundation and Petri net model. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems (ISMIS 2000)*, LNAI 1932, 283–291, Springer, Berlin, 2000.
36. J.F. Peters, A. Skowron, L. Han, S. Ramanna. Towards rough neural computing based on rough membership functions: Theory and application. In *[74]*, 604–611, 2001.
37. J.F. Peters, L. Han, S. Ramanna. Rough neural computing in signal analysis. *Computational Intelligence*, 1(3): 493–513, 2001.
38. L. Polkowski, A. Skowron. Approximate reasoning about complex objects in distributed systems: Rough mereological formalization. In W. Pedrycz, J.F. Peters, editors, *Computational Intelligence in Software Engineering. Advances in Fuzzy Systems-Applications and Theory 16*, 237–267, World Scientific, Singapore, 1998.
39. L. Polkowski, A. Skowron. Rough mereology: A new paradigm for approximate reasoning. *International Journal Approximate Reasoning*, 15(4): 333–365, 1996.
40. L. Polkowski, A. Skowron. Calculi of granules based on rough set theory: Approximate distributed synthesis and granular semantics for computing with words. In *[73]*, 20–28, 1999.
41. L. Polkowski, A. Skowron. Rough-neuro computing. In *[74]*, 57–64, 2001.
42. L. Polkowski, A. Skowron. Towards adaptive calculus of granules. In *Proceedings of the 6th International Conference on Fuzzy Systems (FUZZ-IEEE'98)*, 111–116, Anchorage AK, 1998.
43. 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.
44. A. Skowron. Toward intelligent systems: Calculi of information granules. *Bulletin of the International Rough Set Society*, 5(1/2):9–30, 2001.
45. A. Skowron. Approximate reasoning by agents in distributed environments. In N. Zhong, J. Liu, S. Ohsuga, J. Bradshaw, editors, *Intelligent Agent Technology: Research and Development. Proceedings of the 2nd Asia-Pacific Conference on IAT (APCIAT 2001)*, 28–39, World Scientific, Singapore, 2001.
46. A. Skowron. Approximate reasoning by agents. In *Proceedings of the 2nd International Workshop of Central and Eastern Europe on Multi-Agent Systems (CEEMAS 2001)*, LNAI 2296, 3–14, Springer, Berlin, 2002.
47. 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.
48. A. Skowron, J. Stepaniuk. Information granules in distributed environment. In *[73]*, 357–365, 2001.
49. A. Skowron, J. Stepaniuk, S. Tsumoto. Towards discovery of information granules. In *Proceedings of the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'99)*, LNAI 1704, 542–547, Springer, Berlin, 1999.

50. A. Skowron, J. Stepaniuk. Tolerance approximation spaces. *Fundamenta Informaticae*, 27: 245–253, 1996.
51. A. Skowron, J. Stepaniuk. Information granules and approximation spaces. In *Proceedings of the 7th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'98)*, 1354–1361, Paris, 1998.
52. A. Skowron, J. Stepaniuk. Information granules: Towards foundations of granular computing. *International Journal of Intelligent Systems*, 16(1): 57–104, 2001.
53. A. Skowron, J. Stepaniuk. Information granule decomposition. *Fundamenta Informaticae*, 47(3/4): 337–350, 2001.
54. A. Skowron, J. Stepaniuk, J.F. Peters. Approximation of information granule sets. In *[74]*, 65–72, 2001.
55. A. Skowron, J. Stepaniuk, J.F. Peters. Hierarchy of information granules. In H.D. Burkhard, L. Czaja, H.S. Nguyen, P. Starke, editors, *Proceedings of the Workshop on Concurrency, Specification and Programming (CSP 2001)*, 254–268, Warsaw, 2001.
56. A. Skowron, Z. Suraj. A parallel algorithm for real-time decision making: A rough set approach. *Journal of Intelligent Information Systems*, 7: 5–28, 1996.
57. R.W. Swiniarski. *RoughNeuralLab, software package*. Developed at San Diego State University, San Diego, CA, 1995.
58. R. Swiniarski, F. Hunt, D. Chalvet, D. Pearson. Prediction system based on neural networks and rough sets in a highly automated production process. In *Proceedings of the 12th System Science Conference*, Wrocław, Poland, 1995.
59. R. Swiniarski, F. Hunt, D. Chalvet, D. Pearson. Intelligent data processing and dynamic process discovery using rough sets, statistical reasoning and neural networks in a highly automated production systems. In *Proceedings of the 1st European Conference on Application of Neural Networks in Industry*, Helsinki, 1995.
60. R.W. Swiniarski. Rough sets and neural networks application to handwritten character recognition by complex Zernike moments. In *Proceedings of the 1st Internatonal Conference on Rough Sets and Current Trends in Computing (RSCTC'98)*, LNAI 1424, 617–624, Springer, Berlin, 1998.
61. R.W. Swiniarski, L. Hargis. Rough sets as a front end of neural networks texture classifiers. *Neurocomputing: An International Journal*, 36: 85–103, 2001.
62. M. S. Szczuka. Refining classifiers with neural networks. *International Journal of Intelligent Systems*, 16(1): 39-55, 2001.
63. 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.
64. M.S. Szczuka. Function approximation by neural networks with application of rough set methods. Master's thesis, Faculty of Mathematics, Informatics and Mechanics, Warsaw University, 1995 (in Polish).
65. M.S. Szczuka. Symbolic methods and artificial neural networks in classifier construction, Ph.D. dissertation, Faculty of Mathematics, Informatics and Mechanics, Warsaw University, 2000 (in Polish).
66. M.S. Szczuka. Rough set methods for constructing artificial neural networks. In B.D. Czejdo, I.I. Est, B. Shirazi, B. Trousse, editors, *Proceedings of the 3rd Biennial Joint Conference on Engineering Systems Design and Analysis (ESDA'96)*, 9–14, Montpellier, France, 1996.
67. A. Tarski. In *Introduction to Logic and to the Methodology of Deductive Sciences*, IV, 68–78. Oxford University Press, New York, 1965.
68. N. Wiener. *Cybernetics or Control and Communication in the Animal and the Machine*, 2nd ed. MIT Press, Cambridge, MA, 1961.

69. 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.
70. L.A. Zadeh. Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, 4: 103–111, 1996.
71. L.A. Zadeh. A new direction in AI: Toward a computational theory of perceptions. *AI Magazine*, 22(1): 73–84, 2001.
72. T. Terano, T. Nishida, A. Namatame. S. Tsumoto, Y. Ohsawa, T. Washio, editors. *New Frontiers in Artificial Intelligence. Joint JSAI 2001 Workshop Post Proceedings*, LNAI 2253, Springer, Berlin, 2001.
73. N. Zhong, A. Skowron, S. Ohsuga, editors. *New Directions in Rough Sets, Data Mining, and Granular–Soft Computing*, LNAI 1711, Springer, Berlin, 1999.
74. W. Ziarko, Y.Y. Yao, editors. *Proceedings of the 2nd International Conference on Rough Sets and Current Trends in Computing (RSCTC 2000)*, LNAI 2005, Springer, Berlin, 2001.