Rough Neurocomputing: A Survey of Basic Models of Neurocomputation

James F. Peters[1] and Marcin S. Szczuka[2]

[1] Department of Electrical and Computer Engineering,
University of Manitoba, Winnipeg, Manitoba R3T 5V6, Canada
jfpeters@ee.umanitoba.ca
[2] Institute of Mathematics, Warsaw University
Banacha 2, 02-097, Warsaw, Poland
szczuka@mimuw.edu.pl

Abstract. This article presents a survey of models of rough neurocomputing that have their roots in rough set theory. Historically, rough neurocomputing has three main threads: training set production, calculus of granules, and interval analysis. This form of neurocomputing 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. This work has led to a variety of new rough neurocomputing computational models that are briefly presented in this article. The contribution of this article is a survey of representative approaches to rough neurocomputing.

Keywords. Information granule, mereology, neural network, rough sets.

1 Introduction

The hint that rough set theory provides a good basis for neurocomputing can be found in a discussion about machine learning by Zdzisław Pawlak in 1991 [1]. Studies of neural networks in the context of rough sets [4,5] and granular computing [6] are extensive. The first comprehensive, encyclopedic presentation of rough neurocomputing theory and numerous case studies appears in [5]. Other good sources of intensive studies of rough neurocomputing appear in [4,5].

Rough neurocomputing has three main threads: training set production, calculus of granules, and interval analysis. The first thread of rough neurocomputing, namely, rough set philosophy focuses on inductive learning and the production of training sets using knowledge reduction algorithms. This first thread has a strong presence in current rough neurocomputing research, and leads to a rough set approach to preprocessing that provides input to various forms of neural networks (see, e.g., [6]). The second thread in rough neurocomputing has two main components: information granule construction in distributed systems of agents and local parameterized approximation spaces (see, e.g., [6]). A formal treatment of the hierarchy of relations of being a part in a degree (also known as approximate rough mereology) was introduced by Polkowski and Skowron

J.J. Alpigini et al. (Eds.): RSCTC 2002, LNAI 2475, pp. 308–315, 2002.

in the mid- and late-1990s [9]. 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 instead of vectors of weights commonly used in conventional neural networks. That is, 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 latter part of the 1990s (see, e.g., [4,5,8]).

Practical applications of rough neurocomputing have recently been found in predicting urban highway traffic volume, speech analysis, classifying the waveforms of power system faults, signal analysis, assessing software quality, control of autonomous vehicles, line-crawling robot navigation, EEG analysis, and handwriting recognition (see, e.g., [4,5,7,8]). In its most general form, rough neurocomputing provides a basis for granular computing. A rough mereological approach to rough neural network springs from an interest in knowledge synthesized (induced) from successive granule approximations performed by neurons [6].

This article is organized as follows. A rough set approach to preprocessing in the preparation of inputs to various forms of neural networks is presented in Section 1. An overview of a granular approach to rough neurocomputing is presented in Section 3. A number of different forms of neurons are briefly described in Section 4. The architectures of hybrid forms of neural networks are briefly described in Section 5.

2 Preprocessing with Rough Sets

Rough sets provide symbolic representation of data and the representation of knowledge in terms of attributes, information tables, semantic decision rules, rough measures of inclusion and closeness of information granules, and so on. By contrast, traditional neural networks in their basic form do not consider the

detailed meaning of knowledge gained in the process of model construction and learning. Rather, the focus until recently has been on polynomial approximation in neural computing. In what follows it is assumed that the reader is familiar with the fundamentals of rough set theory [1] and with basic concepts in neurocomputing. This section considers how one can incorporate both approaches (rough sets and neural computing) in a combined system.

Rough set methods make it possible to reduce the size of a dataset by removing some of the attributes while preserving the partitioning of the universe of an information system into equivalence classes. We may consider the possibility of reducing the dataset during preprocessing and then performing construction and learning by a neural network. Let $DT = (U, A, \{d\})$, where U is a finite, non-empty set of objects (universe), A is a set of condition attributes such that for $a \in A$, $X \subseteq U$, $a : X \to V_a$ (value set), and d is a decision attribute. Recall that the indiscernibility relation IND is an equivalence relation such that

$$IND_{DT}(B) = \{(x, x') \in U^2 | \quad \forall_{a \in B \subseteq A} a(x) = a(x')\}$$

Further, a reduct is minimal set of attributes $B \subseteq A$ such that $IND_{DT}(B) = IND_{DT}(A)$, i.e., B preserves the indiscernibility relation [8]. A high-level description of the basic steps in a rough set approach to preprocessing is given in the following algorithm.

Algorithm [Rough Set Approach to Preprocessing]

Input Decision table $DT = (U, A, \{d\})$, where U is a finite, non-empty set of objects (universe), A is a set of condition attributes such that for $a \in A$, $X \subseteq U$, $a : X \to V_a$ (value set), and d is a decision attribute.
Output Reduced table $DT_{reduced}$, calibrated neural network NN
Step 1 Using A, find set of possibly shortest reducts.
Step 2 Reduce DT using some reduct or union of several reducts to create reduced table $DT_{reduced}$, i.e., remove from DT attributes not belonging to the union of selected reducts.
Step 3 Construct a neural network NN over $DT_{reduced}$
Step 4 Calibrate NN from Step 3
Step 5 Repeat Steps 3-4 until sufficient classification accuracy is achieved.
Step 6 Repeat Steps 2-5 until sufficient quality is obtained, then STOP

This algorithm has proven to be very effective for some datasets. Two constraints should be considered in a rough set approach to preprocessing. First, finding a minimal reduct is NP-hard. Hence, it is helpful to use different approximating techniques to find a set of reducts. Second, real-valued attributes can have a very large set of possible values. The solution to this problem is to attempt to reduce the size of an attribute value set.

3 Granular Neural Network Architecture

In this section, the fulfillment of an ontology of approximate reason in a neural network stems from a consideration of granular computing in the context

of parameterized approximation spaces as a realization of an adaptive granule calculus [9]. This realization is made possible by the introduction of a parameterized approximation space in the design of a reasoning system for an agent. A step towards the realization of an adaptive granule calculus in a rough neurocomputing scheme is described in this section and is based on [6]. 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.

Definition 1. *Parameterized Approximation Space. A parameterized approximation space is a system* $AS_{\#,\$} = (U, I_{\#}, R, \nu_{\$})$ *where #, $ denote vectors of parameters, U is a non-empty set of objects and*

- $I_{\#} : U \rightarrow \quad \mathcal{P}(U)$ *is an uncertainty function, where* $\mathcal{P}(U)$ *denotes the powerset of U.*
- $\nu_{\$} : \mathcal{P}(U) \times \mathcal{P}(U) \rightarrow [0, 1]$ *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 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. It should also be noted that for some problems it is convenient to define an uncertainty set function of the form $I_{\#} : \mathcal{P}(U) \rightarrow \quad \mathcal{P}(U)$. 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 $\mathrm{AS}_{\#,\$}$ and any subset $X \subseteq U$, the lower and upper approximations of X in U are defined as follows.

$$LOW\left(AS_{\#,\$}, X\right) = \{x \in U : \nu_{\$}\left(I_{\#}\left(x\right), X\right) = 1\} \text{ [lower approximation]}$$

$$UPP\left(AS_{\#,\$}, X\right) = \{x \in U : \nu_{\$}\left(I_{\#}\left(x\right), X\right) > 0\} \text{ [upper approximation]}$$

Using rough inclusion, the neighborhood $\mathrm{I}_{\#}(\mathrm{x})$ can usually be defined as a collection of close objects. Sets of objects that are collections of objects from a data table are examples of information granules. A parameterized approximation space can be treated as an analogy to a neural network weight. The parameters of an approximation space should be learned to induce the relevant information granules.

4 Rough Neurons

The term *rough neuron* was introduced in 1996 by Lingras [8]. 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 [5]. This has led to the introduction of approximation neurons [5] and their application in classifying electrical power system faults, signal analysis, and in assessing software quality (see, e.g., [4,5]). More recent work on rough measures [3] has led to improved designs of approximation neurons (see, e.g, [5]). An information granulation model of a rough neuron was introduced by Skowron and Stepaniuk in the late 1990s (see, e.g., exposition in [5]). 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 (see, e.g., [4]). The study of various forms of rough neurons is part of a growing number of papers on neural networks based on rough sets.

4.1 Interval-Based Rough Neuron

An interval-based rough neuron was introduced in 1996 [8]. 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 described by a finite set of features (e.g., amplitude, type-of-waveform, high frequency component, rain fall, 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 class of objects based on the values of some features of objects belonging to a class. Let x be a feature variable in the description of an object. Further, let $\overline{x}, \underline{x}$ represent upper and lower bounds of x. In a rough pattern, the value of each feature variable x is specified with $\overline{x}, \underline{x}$ (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 input from neuron i, output from neuron j, and strength of 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 1.

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

The subscript $i = \underline{r}$ for input to a lower neuron, and $i = \overline{r}$ for 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 (2) and (3), respectively.

$$out_{\bar{r}} = \max(t(in_{\bar{r}}), t(in_{\underline{r}})) \quad (2)$$

$$out_{\underline{r}} = \min(t(in_{\bar{r}}), t(in_{\underline{r}})) \quad (3)$$

The output of the rough neuron will be computed in form (4).

$$rough\ neuron\ output = \frac{out_{\bar{r}} - out_{\underline{r}}}{average(out_{\bar{r}}, out_{\underline{r}})} \quad (4)$$

4.2 Approximation Neurons

This section considers the design of rough neural networks containing neurons that perform set approximations and measure rough inclusion, and hence this form of network is called an approximation neural network (aNN). This section is limited to a brief description of one type of neuron in an aNN, namely, approximation neurons (aNs). The architecture of aNs is described in detail in [5]. Preliminary computations in an aN are carried out with a layer of aNs, which construct rough sets and where the output of each aN is a measurement of rough inclusion. Let B, F, $(newObj \cup F)_{B_{approx}}$, $[\mathrm{f}]_B$ denote set of attributes, finite set of neuron inputs (this is an archival set representing past stimuli, a form of memory accessible to a neuron), non-empty finite set of new neural stimuli, set approximation, and equivalence class containing measurements derived from known objects, respectively. Further, the output of aN i.e. the degree of overlap between $(newObj \cup F)_{B_{approx}}$ and $[\mathrm{u}]_B$ is measured using (5).

$$\mu_u^B(newObj \cup F_{B_{approx}}) = \frac{\mathrm{Card}(\ [\mathrm{u}]_B \cap (newObj \cup F)_{B_{approx}})}{\mathrm{Card}(\ [\mathrm{u}]_B)} \quad (5)$$

Other forms of rough neurons are described in [[4,5].

5 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, evolutionary rough-fuzzy MLP, interval-based rough-fuzzy networks, interval-based fuzzy-rough networks, and approximation rough-fuzzy networks (see, e.g., [4,5]). It should also be mentioned that it is common to use rough set theory as a basis for preprocessing inputs to a neural network . In this

section, one form of hybrid network is briefly described: a rough-fuzzy multi-layer perceptron neural network. The Rough-fuzzy MLP (Multi Layer Perceptron) 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, while rough sets extract the domain knowledge for determining the network parameters. The first step in the design of a rough-fuzzy MLP is establish a basis for working with real-valued attribute tables of fuzzy membership values. The traditional model of a discernibility matrix is replaced by (6).

$$c_{ij} = \{a \in B | \, |a(x_i) - a(x_i)| > Th\} \tag{6}$$

for $i, j = 1, ..., n_k$, where Th is an adaptive threshold. Let a_1, a_2 correspond to two membership functions (attributes) with a_2 being steeper as compared to a_1. 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. Here lies the novelty of the proposed method. Moreover, this type of thresholding also enables the discernibility matrix to contain all the representative points/clusters present in a class. This is particularly useful in modeling multi-modal class distributions.

6 Concluding Remarks

Various approaches to rough neurocomputing have been presented in this article. 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 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.

Acknowledgements. The research of James Peters has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) research grant 185986, a grant from Manitoba Hydro, research support from the University of Information Technology and Management, Rzeszów, Poland. The work of Marcin Szczuka has been supported by grant 8T11C02519 from State Committee for Scientific Research and by the Wallenberg Foundation in frame of the WITAS project.

References

1. Z. Pawlak, Rough Sets: Theoretical Aspects of Reasoning About Data. Boston, MA, Kluwer Academic Publishers, 1991.

2. Z. Pawlak, A. Skowron, Rough membership functions. In: R. Yager, M. Fedrizzi, J. Kacprzyk (Eds.), Advances in the Dempster-Shafer Theory of Evidence, NY, John Wiley & Sons, 1994, 251–271.
3. 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 (Eds.), Rough Set Theory and Granular Computing, Bulletin of the International Rough Set Society, vol. 5, no. 1 / 2, 2001, 177–184.
4. S.K. Pal, W. Pedrycz, A. Skowron, R. Swiniarski (Guest Eds.), Neurocomputing: An International Journal, vol. 36, Feb. 2001.
5. S.K. Pal, L. Polkowski, A. Skowron (Eds.), Rough-Neuro Computing: Techniques for Computing with Words. Berlin: Springer-Verlag, 2002.
6. A. Skowron, Toward intelligent systems: Calculi of information granules. In: S. Hirano, M. Inuiguchi, S. Tsumoto (Eds.), Bulletin of the International Rough Set Society, vol. 5, no. 1 / 2, 2001, 9–30.
7. M. S. Szczuka, Rough sets and artificial neural networks. In: L. Polkowski, A. Skowron (Eds.), Rough Sets in Knowledge Discovery 2: Applications, Cases Studies and Software Systems. Berlin: Physica Verlag, 1998, 449–470.
8. P.J. Lingras, Rough neural networks. In: Proc. of the 6^{th} Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU'96), Granada, Spain, 1996, 1445–1450.
9. L. Polkowski, A. Skowron, Towards adaptive calculus of granules. In: Proc. of the Sixth Int. Conf. on Fuzzy Systems (FUZZ-IEEE'98), Anchorage, Alaska, 4-9 May 1998, 111–116.