Granular Computing, logic aspects of bioinformatics, information granules, fuzzy sets, rough sets

Witold PEDRYCZ*

MODELS OF COMPUTATIONAL INTELLIGENCE IN BIOINFORMATICS

Computational Intelligence has emerged as a synergistic environment of Granular Computing (including fuzzy sets, rough sets, interval analysis), neural networks and evolutionary optimisation. This symbiotic framework addresses the needs of system modelling with regard to its transparency, accuracy and user friendliness. This becomes of paramount interest in various modelling in bioinformatics especially when we are concerned with decision-making processes. The objective of this study is to elaborate on the two essential features of CI that is Granular Computing and the resulting aspects of logic-oriented processing and its transparency. As the name stipulates, Granular Computing is concerned with processing carried out at a level of coherent conceptual entities – information granules. Such granules are viewed as inherently conceptual entities formed at some level of abstraction whose processing is rooted in the language of logic (especially, many valued or fuzzy logic). The logic facet of processing is cast in the realm of fuzzy logic and fuzzy sets that construct a consistent processing background necessary for operating on information granules. Several main categories of logic processing units (logic neurons) are discussed that support aggregative (and-like and or-like operators) and referential logic mechanisms (dominance, inclusion, and matching). We show how the logic neurons contribute to high functional transparency of granular processing, help capture prior domain knowledge and give rise to a diversity of the resulting models.

1. INTRODUCTION

The quest for designing and deploying intelligent systems in bioinformatics has been with the community of researchers and practitioners for at least several decades. It manifested in different ways and has been realized in the variety of conceptual frameworks with each of them promoting some research agenda and focusing on some philosophical, methodological and algorithmic aspects (such as pattern recognition, symbolic processing, evolutionary mechanisms). One of the recent developments concerns Computational Intelligence (CI) – a unified and comprehensive platform of conceptual and computing endeavours of fuzzy sets, neurocomputing and evolutionary methods. CI promotes and dwells on synergy: it carefully identifies the nature of the individual technologies and exploits their highly complementary character.

While fully acknowledging the benefits and rationale behind CI, in this study, we elaborate on a different point of view that tackles the fundamentals of the development of intelligent systems. Central to our main line of thought is Granular Computing [1][21] – a cohesive and coherent platform of representing and processing information granules. Such granules are represented in many different ways including sets (intervals), fuzzy sets, and rough sets, to name several dominant alternatives. As being abstract entities, their processing is very much embedded in the framework of logic that is highly consistent with the underlying nature of information granules and supports the transparency of the underlying processing and its semantics.

* Department of Electrical & Computer Engineeering, University of Alberta, Edmonton Canada T6R 2G7, [pedrycz@eece.ualberta.ca], Systems Research Institute, Polish Academy of Sciences, Warsaw Poland

The organization of the material is structured into six sections. First, in Section 2 we discuss the fundamentals of Granular Computing and outline some historical aspects of the main developments along with a brief view at the diversity of the abstraction mechanisms of fuzzy sets, rough sets, and interval analysis. Section 3 is concerned with the unified processing environment of Granular Computing and logic, especially their continuous (multivalued and fuzzy) variants. Subsequently, in Section 4 and 5 we proceed with the realization of the logic layer of this environment by studying various fuzzy logic neurons and networks. Conclusions are covered in Section 6.

2. GRANULAR COMPUTING

Information granules permeate human endeavours. No matter what specific task is taken into account, we usually cast it into a certain conceptual setting. This is the framework we formulate generic concepts adhering to some appropriate level of abstraction, carry out further processing, and communicate the results to the environment. Information granules are pivotal to a way in which experimental datasets are perceived and interpreted by humans. Instead of being buried in huge piles of numeric data, the user summarizes such datasets and forms a very limited vocabulary of information granules (for instance, fuzzy sets) that are easily comprehended, come with a well-defined semantics and in this manner help express main relationships and dependencies between individual variables arising at some specific level of information granularity. Even more prominently, granularity implements information abstraction: by moving to a certain level of abstraction (granulation) we hide unnecessary details and concentrate on essentials (main dependencies, general trends) existing in the data. Quite frequently, information granules are formed via data clustering.

Granular Computing is an emerging and highly unifying paradigm of information processing. While we have already noticed a number of important conceptual and computational constructs developed in the domain of system modelling, machine learning, image processing, pattern recognition, and data compression in which various abstractions (and ensuing information granules) came into existence, Granular Computing has become innovative and intellectually proactive in several fundamental ways. It helps identify the essential commonalities between the surprisingly diversified problems and technologies used there, which could be cast into a unified framework we usually refer to as a granular world. This is a fully operational processing entity that interacts with the external world (that could be another granular or numeric world) by collecting necessary granular information and returning the outcomes of the granular computing. With the emergence of the unified framework of granular processing, we come up a better grasp as to the role of interaction between various formalisms and visualize mechanisms of interaction between them. It brings together the existing formalisms of set theory (interval analysis), fuzzy sets, rough sets, etc. under the same roof by clearly visualizing that in spite of their visibly quite distinct underpinnings (and ensuing processing), they also come with fundamental commonalities. In this sense, Granular Computing establishes a stimulating environment of synergy between the individual approaches.

By dwelling on the commonalities of the existing formal approaches, Granular Computing helps build heterogeneous and multifaceted models of processing of information granules by clearly recognizing the orthogonal nature of some of the existing and well established frameworks (say,

probability, namely probability density functions and fuzzy sets).Granular Computing fully acknowledges a notion of variable granularity whose range could cover detailed numeric entities and very abstract and general information granules. It looks at the aspects of compatibility of such information granules and ensuing communication mechanisms of the granular worlds.

With the fundamentals of Granular Computing come a variety of conceptual frameworks and ensuing algorithms. These are quite different from the standpoint of the methods, interpretation of results, and interaction between other environments of Granular Computing. The diversity is quite profound; here we briefly comment on some of them and elaborate on the most essential features. By no means we pretend to cover the topic in detail; with the number of authoritative publications in the area is not necessary. What becomes perhaps far more needed is a synthetic and comparative analysis of various alternatives of Granular Computing.

Historically, the first framework of Granular Computing was realized in the language of sets and intervals (giving rise to so-called interval analysis). The origin of this interval calculus or interval analysis was rooted in the realm of finite precision numeric computing. These highly conservative models help express important phenomena of propagation and accumulation of computing errors. As the bounded yet finite granules, intervals were quite attractive in describing numerous phenomena ranging from system identification, control under uncertainty, and pattern classification. The concept of fuzzy sets coined in 1965 was critical in re-defining the fundamental idea of dichotomy (yes-no, membership-exclusion, etc.) sets or intervals fully subscribed to. Since then fuzzy sets introduced and supported a notion of partial membership of element to a given concept (information granule), cf. [24][25]. The applied facets of fuzzy sets is represented across many disciplines including control and decision-making, cf.[10][20], intelligent systems [11], and pattern recognition [14]. Rough sets [15] allow capture concepts with “rough” boundaries where there we come up with a description of concepts articulated in terms of lower and upper boundaries. Such boundaries arise naturally when we attempt to articulate a complex concept in the language of less descriptive, simpler entities that form our basic vocabulary. Rough sets and fuzzy sets are complementary to a significant extent; we have witnessed emergence of hybrid granules in the form of rough-fuzzy or fuzzy-rough sets. Further generalized concepts of information granules can be found in [1][2][3].

Defining and subsequently measuring the size of information granules is of apparent relevance to Granular Computing and particular models developed there. We can come up with a uniform view on this matter in spite of the striking diversity of the underlying formalisms. Intuitively, the size of the granules relates in one way or another to the number of elements they comprise of. The terms such as set cardinality, width of an interval, σ-count of fuzzy set come immediately to mind. They are especially appealing and intuitively meaningful as they relate to the very essence of abstraction: the unified treatment and a cohesiveness of elements drawn together by the information granule under discussion.

3. GRANULAR COMPUTING AND LOGIC: SYNERGISTIC LINKS

In essence, Granular Computing promotes the concept of abstraction – a fundamental faculty of human endeavours. The level of abstraction is inherently problem and user oriented. Different levels of abstraction that reflect upon the detail of problem description in terms of basic concepts

(entities), primary relationships between them, interaction with the user/analyst. As driven by abstraction, the resulting variables and relationships between them (as being perceived at the particular level of detail) are more abstract entities. This naturally implies that their processing should exploit the language that is in par with the fundamentals of such information granules. Casting the processing in logic comes as a viable and highly justifiable alternative. What type detailed logic is deemed the most appropriate arises as the next specific question. Likewise in Granular Computing, we have at our disposal a variety of logics starting from two-valued one (leading to the well known Boolean calculus, digital systems, etc.), moving to three-valued logic and multivalued logics and including fuzzy logic. The option of fuzzy logic arises here as an interesting alternative owing to the richness of the underlying semantics and useful computing facilities implied primarily by the truth values located in the unit interval.

In a nutshell, the proposed logic kernel (called here fuzzy networks) completes a logic-based processing of input signals and realizes a certain logic-driven mapping between input and output spaces. As the networks interact with a physical world whose manifestation does not usually arise at the level of logic (multivalued) signals, it becomes apparent that there is a need for some interface of the model. Such interfaces are well known in fuzzy modelling [20], cf. also [13][18][19]. In the literature they commonly arise under a name of fuzzifiers (granular coders aimed at information granulation) and defuzzifiers (granular decoders with a main function of degranulation of information). The role of the coder is to convert a numeric input coming from the external environment into the internal format of membership grades of the fuzzy sets defined for each input variable. These results of a non-linear normalization of the input (no matter what original ranges the input variables assume) and a linear increase of the dimensionality of the new logic space in comparison with the original one). On the other hand, the decoder accepts the results of the logic processing and transforms them into some numeric values. The layered architecture of the fuzzy models with clearly distinguished interfaces and the logic-processing core is illustrated in Figure 1.

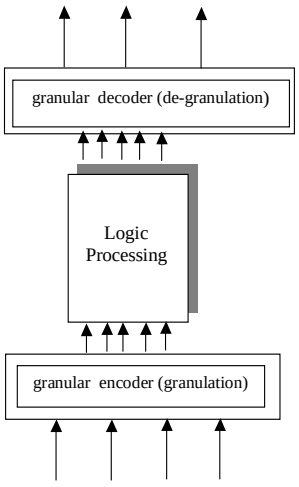


Fig.1. A general layered structure of fuzzy modelling; the use of granular encoders and decoders are essential in the development of communication mechanisms with the modelling environment

With the design of the interfaces, we exercise two general approaches (in the literature we encounter far more diversified techniques but they are usually more specialized in the sense of the underlying computing mechanisms):

(a) granulation of individual variables This mechanism of granulation is quite common in the realm of fuzzy modelling. In essence, we define several fuzzy sets in the universe of discourse of the variable of interest so that any input is transformed via the membership functions defined there and the resulting membership grades are used in further computations by the model. From the design standpoint, we choose a number of fuzzy sets, type of membership functions and a level of overlap between consecutive fuzzy sets. Some general tendencies along this line are thoroughly reported in the literature. By selecting the number of fuzzy sets (usually between 3 and 9 with the magic of 7 ± 2 emphasized throughout the literature), we position modelling activities at some required level of information granularity (a level of modelling details we are interested in). The type of membership functions helps model the semantics of the information granules. Among many possibilities, we commonly encounter triangular fuzzy sets and Gaussian membership functions. These two types come with an extensive list of arguments that help make a suitable selection with respect to the main objectives of the model (e.g., those concerning a trade-off between interpretation and accuracy of modelling). The overlap level is essential from different points of view, namely (a) semantics of the linguistic terms, (b) non-linear numeric characteristics of the fuzzy model, and (c) completeness of the model.

(b) nonlinear or linear normalization Here we transform an original variable defined in some space, say [a,b] (subset of R) is scaled to the unit interval. This could be done with the aid of some mapping ϕ: [a,b] $\rightarrow$ [0,1] that could be either linear or non-linear. In any case we consider that f is monotonically increasing with $\phi(a) = 0$ $\phi(b) = 1$. This transformation does not affect the dimensionality of the problem.

4. MAIN CATEGORIES OF FUZZY LOGIC PROCESSING UNITS

In this section, we present the main categories of the logic neurons as they were introduced and discussed in [22] [24][25]. The underlying taxonomy involves aggregative and referential neurons and very much ties up with their logic functionality. The naming of the neurons reflect the underlying processing realized by them. The aggregative neurons concentrate on the logic type of aggregation of the inputs (truth values) while the referential neurons are aimed at logic processing of results of referential transformations of the corresponding truth values.

4.1. AGGREGATIVE NEURONS

Formally, these neurons realize a logic mapping from $[0,1]^n$ to [0,1]. Two main classes of the processing units exist in this category.

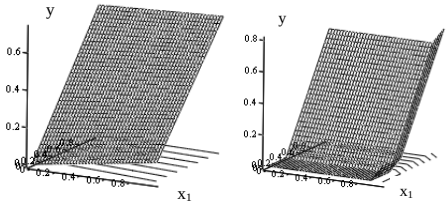
OR neuron realizes an *and* logic aggregation of inputs $x = [x_1, x_2,\ldots,x_n]$ with the corresponding connections (weights) $w = [w_1, w_2,\ldots, w_n]$ and then summarizes the partial results in an *or*-wise manner (hence the name of the neuron). The concise notation underlines this flow of computing, y = OR(x; w) while the realization of the logic operations gives rise to the expression (we commonly refer to it as an s-t convolution)

$$y = \sum_{i=1}^{n} (x_i \, t \, w_i)$$

Bearing in mind the interpretation of the logic connectives (t- and s-norms), the OR neuron realizes the following logic expression carried out on the input signals

$$(x_1 \text{ and } w_1) \text{ or } (x_2 \text{ and } w_2) \text{ or } \ldots \text{ or } (x_n \text{ and } w_n)$$

Apparently the inputs are logically "weighted" by the values of the connections before producing the final result. In other words we can treat "y" as a truth value of the above statement where the truth values of the inputs are affected by the corresponding weights. Noticeably, lower values of w_i discount the impact of the corresponding inputs; higher values (especially those being positioned close to 1) do not affect the original truth values of the inputs resulting in the logic formula. In limit, if all connections w_i , i=12,…,n are set to 1 then the neuron produces a plain *or*-combination of the inputs, $y = x_1$ *or* x_2 *or* … *or* x_n. The values of the connections set to zero eliminate the corresponding inputs. Computationally, the OR neuron exhibits non-linear characteristics (that is inherently implied by the use of the t- and s-norms (that are evidently non-linear mappings). The plots of the characteristics of the OR neuron shown in Figure 2 visualize this effect (note that the characteristics are affected by the use of some norms). The connections of the neuron contribute to its adaptive character; the changes in their values form the crux of the parametric learning.



(a) (b)

Fig.2. Characteristics of the OR neuron for selected pairs of t- and s-norms. In all cases the corresponding connections are set to 0.l and 0.7 with intent to visualize their effect on the input-output characteristics of the neuron: (a) product and probabilistic sum, (b) Lukasiewicz AND and OR connectives (b)

AND neuron The neurons in the category, denoted by y = AND(x; w) with x and w being defined as in case of the OR neuron, are governed by the expression

$$y = \prod_{i=1}^{n} (x_i \text{sw}_i)$$

Here the *or* and *and* connectives are put together in a reversed order: first the inputs are combined with the use of the s-norm and the partial results are aggregated *and*-wise. Higher values of the connections reduce impact of the corresponding inputs. In limit w_i =1 eliminates the relevance of x_i. With all w_i set to 0, the output of the AND neuron is just an *and* aggregation of the inputs

$$y = x_1 \text{ and } x_2 \text{ and } \ldots \text{ and } x_n$$

Let us conclude that the neurons are highly non-linear processing units depending upon the specific realizations of the logic connectives. They also come with potential plasticity whose usage becomes critical when learning the networks involving these neurons.

4.2. REFERENTIAL (REFERENCE) NEURONS

The essence of referential computing deals with processing logic predicates. The two-argument (or generally multivariable) predicates such as *similar*, *included in*, *dominates* are essential components of any logic description of a system. In general, the truth value of the predicate is a degree of satisfaction of the expression P(x, a) where "a" is a certain reference value (reference point). Depending upon the meaning of the predicate (P), the expression P(x, a) reads as " x is similar to a", " x is included in a", "x dominates a", etc. In case of many variables, the compound predicate comes in the form $P(x_1, x_2, .., x_n, a_1, a_2, .., a_n)$ or more concisely P(x; a) where x and a are two vectors in the n-dimensional unit hypercube. We envision the following realization of P(x; a)

$$P(x; a) = P(x_1, a_1) \text{ and } P(x_2, a_2) \text{ and } \ldots \text{ and } P(x_n, a_n)$$

meaning that the satisfaction of the multivariable predicate relies on the satisfaction realized for each variable separately. As the variables could come with different levels of relevance as to the overall satisfaction of the predicates, we represent this effect by some weights (connections) w_1, w_2, …, w_n so that the above expression can be given in the following form

$$P(x; a, w) = [P(x_1,a_1) \text{ or } w_1] \text{ and } [P(x_2,a_2) \text{ or } w_2] \text{ and } \ldots \text{ and } [P(x_n,a_n) \text{ or } w_n]$$

Taking another look at the above expression and using a notation $z_i = P(x_i,a_i)$, it converts to a certain AND neuron y =AND(z; w) with the vector of inputs z being the result of the computations done for the logic predicate. Then the general notation to be used reads as REF(x; w, a) and using the explicit notation we have

$$y = \prod_{i=1}^{n} (\text{REF}(x_i, a_i) \text{sw}_i)$$

In essence, as visualized in Figure 3, we may conclude that the reference neuron is a realized in a two-stage construct where first we determine the truth values of the predicate (with a treated as a reference point) and then treat these results as the inputs to the AND neuron.

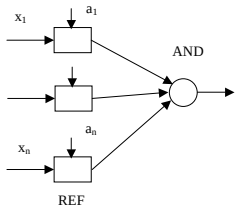


Fig.3. A schematic view of computing realized by a reference neuron an involving two processing phases (referential computing and aggregation)

So far we have exploited the general term of predicate computing not confining ourselves to any specific nature of the predicate. Among a number of possibilities, we discuss the three of them, which tend to occupy an important role. Those are predicates of inclusion, dominance and matching (similarity). As the names stipulate, the predicates return truth values of satisfaction of the relationship of inclusion, dominance and similarity of a certain argument "x" with respect to the given reference "a". The essence of all these calculations is in the determination of the given truth values and this is done in the carefully developed logic framework so that the operations retain their semantics and interpretability. What makes our discussion coherent is the fact that the proposed operations originate from triangular norms. The inclusion operation, denoted by $\subset$ is modelled by an implication $\rightarrow$ that is induced by a certain left continuous t-norm [18][20]

$$a \rightarrow b = \sup\{c \in [0,1] \mid atc \leq b\}, a, b \in [0,1]$$

For instance, for the product the inclusion takes on the form $a \rightarrow b = \min(1, b/a)$. The intuitive form of this predicate is self-evident: the statement "x is included in a" and modelled as INCL(x, a) = x $\rightarrow$ a comes with the truth value equal to 1 if x is less or equal to a (which in other words means that x is included in a) and produces lower truth values once x starts exceeding the truth values of "a". Higher values of "x" (those above the reference point "a") start generating lower truth values of the predicate. The dominance predicate acts in a dual manner. It returns 1 once "x" dominates "a" (so that its values exceeds "a") and values below 1 for x lower than the given threshold. The formal model can be realized as DOM(x, a) = a $\rightarrow$ x. With regard to the reference neuron, the notation is equivalent to the one being used in the previous case (xx), that is DOM(x; w, a) with the same meaning of a and w.

The similarity (match) operation is an aggregate of these two, SIM(x,a) = INCL(x,a) t DOM(x,a) which is appealing from the intuitive standpoint: we say that x is similar to a if x is included in a *and* x dominates a. Noticeably, if x =a the predicate returns 1; if x moves apart from "a" the truth value of the predicate becomes reduced. The resulting similarity neuron is denoted by SIM(x; w, a) and reads as

$$y = \prod_{i=1}^{n} (SIM(x_i, a_i) sw_i)$$

The reference operations form an interesting generalization of the threshold operations in the sense we admit some partial satisfaction of the constraints (boundaries). It is also worth noting that by moving the reference point to the origin (0) and 1-vertex of the unit hypercube (with all its coordinates being set up to 1), the referential neuron starts resembling the aggregative neuron. In particular, we obtain

for a = 1 = [1 1 1… 1] the inclusion neuron reduces to the AND neuron

for a = 0 = [0 0 0… 0] the dominance neuron reduces to the standard AND neuron

One can draw a loose analogy between some types of the referential neurons and the two categories of processing units we encounter in neurocomputing [11]. The analogy is based upon the *local* versus *global* character of processing realized therein. Perceptions come with the global character of processing. Radial basis functions realize a local character of processing as focused on

receptive fields. In the same vein, the inclusion and dominance neurons are after the global nature of processing while the similarity neuron carries more confined, local processing.

5. A GENERAL TOPOLOGY OF THE NETWORK

As we have developed a host of logic processing units, we can use then in the developing a general architecture of the network. In this design we are guided by several requirements. First, we would like to achieve a substantial level of flexibility so that the structure could be easily and effectively adjusted to the experimental data and a variety of problems and their underlying logic settings. Second, we would like to assure a high level of interpretability which is the case here: evidently each neuron comes with a well-defined semantics and our intent is to retain it so at the very end the network can be easily mapped (translated) into a well-structured and transparent logic expression. This quest for interpretability and transparency has been clearly identified and strongly promoted in the most recent literature, cf. [4]; refer also to [5][6][13][20][23] for additional issues raised with this regard. In the logic description we will dwell upon the well-established components of fuzzy logic: logic operators and linguistic modifiers. Having these requirements in mind, a general architecture of the network is shown in Figure 6.

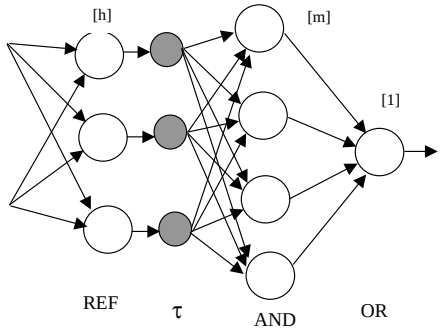


Fig.6. A general architecture of the network constructed with logic-based neurons; see a detailed description in text. The dimensions of the layers (number of nodes) are marked by numbers placed in brackets (upper part of the figure)

The network comes with several layers where each of them has a clear functional role to play and is highly interpretable. The first layer (carrying out some specialized referential processing) is composed of "h" referential neurons (inclusion, dominance, similarity). The results of this processing are raised to some power (indicated by some small shadowed circle) and then combined *and*-wise in the third layer of the network. The elements there are AND neurons with all connections hardwired to zero. The width of this layer is equal to "m". In the sequel the results are combined by the layer of OR neurons. Let us now move on to the computational details by the same time concentrating on the interpretation of the underlying computations. The truth values generated by the referential neurons reflect the level of satisfaction of the logic predicates

$$z_1 = P_1(x; a_1)\ z_2 = P_2(x; a_2), \ldots, z_h = P_h(x; a_h)$$

The powers of z_i, denoted here as τ_i (z_i) where τ_i assumes a few discrete values (say ½, 0, 1, 2, and 4) are interpreted as linguistic modifiers operating upon z_i and producing some concentration or

dilution effect [20][24]. More specifically, the collection of the modifiers maps on the corresponding powers of the truth values in a usual way we commonly encounter in the literature

½ - *more or less* (dilution effect)

0 - unknown

1 – true (neutral)

2 - *very* (concentration effect)

4 - *very* (*very*) = *very* ^{2} (strong concentration effect)

For instance, the expression $\mathrm{INCL}([x_1, x_2], [0.2, 0.7], [0.6, 0.9])^{0.5}$ that involves the two first layers of the network translates into the following linguistic statement

$$y = \text{more or less } \{ [x_1 \text{ included in } 0.6] \text{ or } 0.2 \text{ and } [x_2 \text{ included in } 0.9] \text{ or } 0.7 \}$$

(noticeably, the core part of this expression could be extracted by carrying out some additional pruning). In a similar way, by writing down the logic formulas for the AND and OR neurons, we arrive a complete logic expression of the network.

While we have concentrated on the interpretability of the logic networks, it is equally important to discuss its development. As they fall under the umbrella of specialized neural networks, a lot of gradient-based learning techniques designed there are of immediate use to us and they have been used in the past, cf. [8][11][16][17][18][19]. In contrast, the structure discussed here is highly heterogeneous. In the optimization we are immediately faced with the problem of structural optimization (e.g., when building the first layer of the referential neurons) that is beyond the reach of the popular gradient based learning; various mechanisms of genetic optimization become of interest [7][20].

6. CONCLUSIONS

The diversity of modelling pursuits in bioinformatics has raised an important issues of transparency and interpretability of its models. This aspects of modelling are vitally essential when it comes to decision-making processes and data mining activities. Being motivated by the genuine need of constructing networks that exhibit plasticity while retaining interpretability, we have developed a heterogeneous structure composed of logic neurons. The two main categories of aggregative and reference neurons are deeply rooted in the fundamental operations encountered in fuzzy sets (including logic operations, linguistic modifiers, and logic reference operations). The direct interpretability of the network we addressed in the study helps develop a transparent logic description of data. As the network takes advantage of using various neurons, this imposes an immediate requirement of structural optimization and leads to the utilization of the mechanisms of genetic optimization (genetic algorithms).

BIBLIOGRAPHY

[1] A. BARGIELA, W. PEDRYCZ, *Granular Computing: An Introduction*, Kluwer Academic Publishers, Dordrecht, 2002.

[2] Z. BUBNICKI, Uncertain variables and their application to decision making problems, *IEEE Trans. on Systems, Man, and Cybernetics –A*, 31, no.6, 2001, 587-596.

[3] Z. BUBNICKI, *Uncertain Logics, Variables and Systems*, Springer Verlag, Berlin, 2002.

[4] J. CASILLAS et al. (eds.), *Interpretability Issues in Fuzzy Modeling*, Springer Verlag, Berlin, 2003.

[5] J. A. DICKERSON, M.S. LAN, Fuzzy rule extraction from numerical data for function approximation, *IEEE Trans on System, Man, and Cybernetics -B*, 26, 1995, 119-129.

[6] A.F. GOMEZ-SKARMETA, M. DELGADO, M.A. VILA, About the use of fuzzy clustering techniques for fuzzy model identification, *Fuzzy Sets and Systems*, 106, 1999, 179-188.

[7] E. GOLDBERG, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley,1989

[8] K. HIROTA, W. PEDRYCZ, OR/AND neuron in modeling fuzzy set connectives, *IEEE Trans. on Fuzzy Systems*, 2, 1994, 151-161.

[9] K. HIROTA, W. PEDRYCZ, Fuzzy relational compression, *IEEE Trans. on Systems, Man, and Cybernetics-B*, 29, 1999, 407-415.

[10] J. KACPRZYK, *Wieloetapowe Sterowanie Rozmyte*, PWN, Warszawa, 2001.

[11] B. KOSKO, *Neural Networks and Fuzzy Systems*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[12] S. MITRA, S.K. PAL, Logical operation based fuzzy MLP for classification and rule generation, *Neural Networks*, 7, 1994, 353-373.

[13] S. MITRA, S.K. PAL, Fuzzy multiplayer perceptron, inferencing and rule generation, *IEEE Trans. on Neural Networks*, 6, 1995, 51-63.

[14] S.K. PAL, S. MITRA, *Neuro-Fuzzy Pattern Recognition*, J. Wiley, N. York, 1999.

[15] Z. PAWLAK, *Rough Sets. Theoretical aspects of Reasoning about Data*, Kluwer Academic Publishers, Dordercht, 1991.

[16] W. PEDRYCZ, Neurocomputations in relational systems, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13, 1991, 289-297.

[17] W.PEDRYCZ, Fuzzy neural networks and neurocomputations, *Fuzzy Sets and Systems*, 56, 1993, 1-28.

[18] W. PEDRYCZ, A. ROCHA, Knowledge-based neural networks, *IEEE Trans. on Fuzzy Systems*,1, 1993, 254-266.

[19] W. PEDRYCZ, P. LAM, A.F. ROCHA, Distributed fuzzy modelling, *IEEE Trans. on Systems, Man and Cybernetics-B*, 5, 1995, 769 - 780.

[20] W. PEDRYCZ, F. GOMIDE, *An Introduction to Fuzzy Sets: Analysis and Design*, MIT Press, Boston, 1998.

[21] W. PEDRYCZ (ed.), *Granular Computing. An Emerging Paradigm*, Physica-Verlag, Heidelberg, 2001.

[22] M. SETNES, R. BABUSKA, H. VEBRUGGEN, Rule-based modeling: precision and transparency, *IEEE Trans on System, Man, and Cybernetics - C*, 28, 1998, 165-169.

[23] T. SUDKAMP, R.J. HAMMEL II, Rule base completion in fuzzy models, In: W. Pedrycz (ed.), *Fuzzy Modelling: Paradigms and Practice*, Kluwer Academic Publishers, Dordercht, 1996, pp. 313-330.

[24] L.A. ZADEH, Fuzzy logic-= computing with words, *IEEE Trans. on Fuzzy Systems*, 4, 1996, 103-111.

[25] L.A. ZADEH, Towards a theory of fuzzy information granulation and its application in human reasoning and fuzzy logic, *Fuzzy Sets and Systems*, 90, 1997, 111-127.