Knowledge Transfer in System Modeling and Its Realization Through an Optimal Allocation of Information Granularity

[&]Witold Pedrycz, [*]Barbara Russo, and [*]Giancarlo Succi

[&]Department of Electrical & Computer Engineering, University of Alberta
Edmonton T6R 2G7 AB Canada
and
Systems Research Institute, Polish Academy of Sciences
Warsaw, Poland

[*]Centre for Applied Software Engineering, Faculty of Computer Science, Free University of Bolzano-Bozen, Piazza Domenicani 3, 39100 Bolzano, Italy

Abstract In this study, we introduce and discuss a concept of knowledge transfer in system modeling. In a nutshell, knowledge transfer is about ways on how a source of knowledge (that is an existing model) can be used in presence of new, very limited experimental evidence. As such new data are very scarce; they are not sufficient to construct a new model. At the same time, the new data originate from a similar (but not the same) phenomenon (process) for which the original model has been constructed. Such situations can be encountered in software engineering where in spite existing similarities, each project, process, product exhibits its own unique characteristics. Taking this into consideration, the existing model is generalized (abstracted) by forming its *granular* counterpart- granular model where its parameters are regarded as information granules rather than numeric entities, viz. their non-numeric (granular) version is formed based on the values of the numeric parameters present in the original model. The results produced by the granular model are also granular and in this manner they become reflective of the differences existing between the current phenomenon and the process for which the previous model has been formed.
In the study on knowledge transfer and reusability, information granularity is viewed as an important design asset and as such it is subject to optimization. We formulate an optimal allocation problem: assuming a certain level of granularity, distribute it among the parameters of the model (making them granular) so that a certain data coverage criterion is maximized. While the underlying concept is general and applicable to a variety of models, in this study, we discuss its use to fuzzy neural networks. Several granularity allocation protocols are discussed and their effectiveness is quantified. The use of Particle Swarm Optimization (PSO) as the underlying optimization tool to realize optimal granularity allocation is discussed.

Keywords: knowledge transfer and knowledge reusability, granular model, information granularity, optimal granularity allocation, software cost estimation, Computational Intelligence.

1. Introductory comments: a concept of knowledge transfer

In modeling systems, processes and phenomena, we can regard a resulting model as a source of knowledge. Once being constructed on a basis of some usually quite large experimental data **D**, this source of knowledge can be used afterwards for a variety of prediction, control and description tasks thus contributing to the better understanding of the system. The quality of the model and usefulness depends upon the nature of the new scenarios in which the model is used. In particular, prior to its use it becomes essential to assess how these new situations are different from those manifesting by the data used to construct the model. In many complex problems of planning, cost estimation of software projects, each scenario is quite different. The sources of knowledge (models) formed so far could be useful but must be treated with caution when being applied to new situations. They might be useful but the results require some interpretation.

Let us consider that for a current problem at hand we are provided with a very limited data set – experimental evidence **D'**. Given this small data, two possible scenarios might be envisioned:
(a) we can attempt to construct a model based on the data **D'**. As the current data set is very limited, designing a new model does not look quite feasible: it is very likely that the model cannot be constructed at all, or even if formed, the resulting construct could be of low quality.
(b) we may rely on the existing model (which although deals with not the same situation but has been formed on a large and quite representative body of experimental evidence and we may take advantage of the experience accumulated so far) and augment it in a certain sense so that it becomes adjusted to the current quite limited albeit current data. In doing this, we fully acknowledge that the existing source of knowledge has to be taken with a big grain of salt and the outcomes of the model have to be reflective of partial relevance of the model in the current situation. We quantify this effect by making the parameters of the model granular (viz. more abstract and general) so that one can build the model around the conceptual skeleton provided so far. In this case, viewing the model obtained so far as a sound source of knowledge, we are concerned with a concept of an effective knowledge transfer. The knowledge transfer (which, in essence, is represented by some model denoted here by N) manifests in the formation of a more abstract version of the original model- a so called granular model, G(N) where the granular nature of the model associates with the augmentation (abstraction) of the original model N being realized in presence of new data.

The process of knowledge transfer is intuitively appealing and becomes visible in many endeavors. As a compelling example, consider models of quantitative software engineering [1][2][8][9][10][11][17]. We build models of processes and qualities of software. In software cost estimation, project planning, quality assessment to come up with some models whose construction relies on collected experimental data.

Each software project is unique so the model designed on a basis of the previous data might not be completely relevant however at the same time could not be neglected at all.

Building a model for this specific process or software quality could not be feasible - simply one might have a very limited data set, especially in case of an initial phase of the project or when there have not been substantial efforts to systematically collect data. In light of these, we encounter knowledge transfer – here the available model is viewed only as an initial construct that requires more revising/adjustments.

In general, the essence of the process of knowledge transfer is illustrated in Figure 1. The original model, call it N, built on basis of **D** is now *abstracted* through its granulation, yielding its granular version G(N), and this occurs when dealing with a new data **D'**. The granular model becomes more in rapport with the environment currently encountered. Furthermore the level of granularity is regarded here to be an important design asset whose efficient or optimal allocation helps in an effective usage of knowledge already acquired on a basis of **D**. The allocation itself is regarded as an optimization vehicle to make the model more in rapport with the reality.

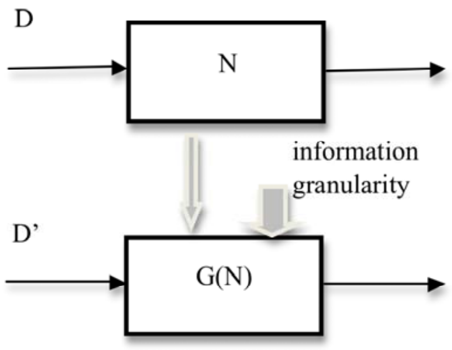


Figure 1. From model (N) to its granular counterpart (G(N)) being a result of the realization of knowledge transfer

The generalization of the effect of knowledge transfer can be discussed in case of "p" different sources of knowledge – models built on a basis of $\mathbf{D}_1$, $\mathbf{D}_2$, …, $\mathbf{D}_p$, say N_1, N_2, …,N_p. We are interested in determining such N_{i0}, which is can be abstracted (granulated) in the most efficient way. In other words, N_{i0} is the one for which $G(N_{i0})$ leads to the best representation (quantified by means of some objective function) among all models available. Denoting this objective function of interest to us by Q, the problem is formulated as an optimization task of the form

$$i_0 = \arg\min_{i=1,2,\ldots,p} Q(G(N_i)) \tag{1}$$

Again as before, a certain level of information granularity becomes available to form a granular version of the original model; refer to Figure 2 highlighting the very concept of knowledge transfer.

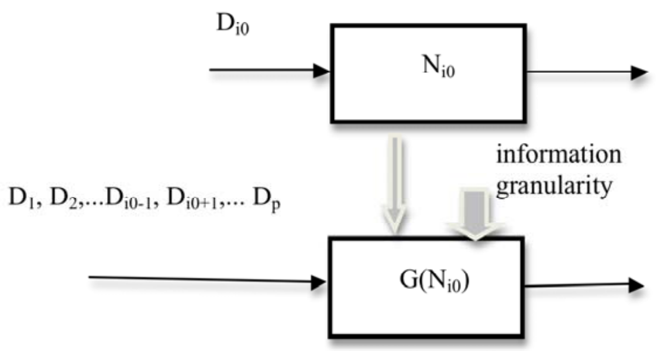


Figure 2. Formation of the best granular model among a family of locally constructed models $N_1, N_2, \ldots, N_p$

2. Fuzzy logic networks – architectural considerations

The constructs of Computational Intelligence bringing together ideas of neurocomputing and fuzzy sets offer a great deal of synergy of resulting constructs. In what follows, we introduce a concept of fuzzy logic networks, elaborate on the underlying architecture as well as their interpretability, and look at some related design practices.

2.1. Realization of a fuzzy logic mapping

Fuzzy sets and information granules, in general, offer a structural backbone of fuzzy logic networks. The crux of the concept is displayed in Figure 3. Information granules [3][4][5][18] $A_1, A_2, \ldots A_c$ are formed in the feature (input) space and output space. The information granules in the input space, $B_1, B_2, \ldots, B_m$ are *logically* associated with the information granules positioned in the output space in the sense that for each of them a degree of activation is a logic function (logic mapping). The flexibility of the logic mapping is offered through the use of the collection of logic neurons (fuzzy neurons) whose connections are optimized during the design process.

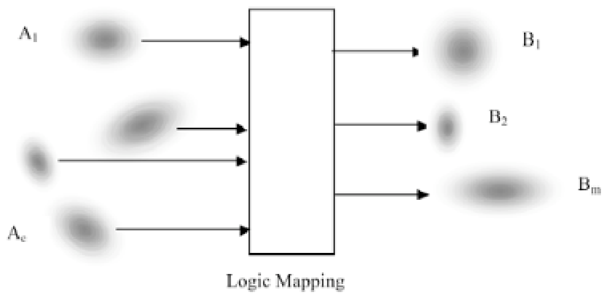


Figure 3. An overall scheme of logic mapping between information granules – fuzzy sets in the input and output space and realized in a form of fuzzy logic network

We start by looking at the functional components of the network– logic neurons.

2.2. Main categories of aggregative fuzzy neurons: AND and OR neurons

Logic neurons [7][14][16] come with a clearly defined semantics of its underlying logic expression and are equipped with significant parametric flexibility necessary to facilitate substantial learning abilities. Formally, a logic neuron realizes a logic mapping from $[0,1]^n$ to $[0,1]$. Two main classes of the processing units are distinguished:

OR neuron: This neuron realizes an *and* logic aggregation of inputs $\mathbf{x} = [x_1\ x_2 \ldots x_n]$ with the corresponding connections (weights) $\mathbf{w} = [w_1\ w_2 \ldots\ w_n]$ 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 (commonly referring to it as an s-t combination or s-t aggregation)

$$y = \mathop{S}_{i=1}^{n}(w_i t x_i) \tag{2}$$

Recall that t- norms and t-conorms (s-norms) are the generic models of logic operators used in fuzzy sets, cf [14] Some commonly encountered examples are the minimum and product (t-norms) and the minimum operator and the probabilistic sum (t-conorms). Bearing in mind the interpretation of the logic connectives (t-norms and t-conorms), the OR neuron realizes the following logic expression being viewed as an underlying logic description of the processing of the input signals

$$(x_1 \textit{ and } w_1) \textit{ or } (x_2 \textit{ and } w_2) \textit{ or } \ldots \textit{ or } (x_n \textit{ and } w_n) \tag{3}$$

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 of the connections (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 =1, 2,…,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 nonlinear characteristics (that is inherently implied by the use of the t- and t-conorms (that are evidently nonlinear mappings). The connections of the neuron contribute to its adaptive character; the changes in their values form the crux of the parametric learning.

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

$$y = \mathop{T}_{i=1}^{n}(w_i s x_i)$$

(4)

Here the *or* and *and* connectives are used in a reversed order: first the inputs are combined with the use of the t-conorm (s-norm) and the partial results produced in this way 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 connections w_i set to 0, the output of the AND neuron is just an *and* aggregation of the inputs

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

(5)

Let us conclude that the neurons are highly nonlinear processing units whose nonlinear mapping depends upon the specific realizations of the logic connectives. They also come with potential plasticity whose usage becomes critical when learning the networks including such neurons.

At this point, it is worth contrasting these two categories of logic neurons with “standard” neurons we encounter in neurocomputing. The typical construct there comes in the form of the weighted sum of the inputs $x_1, x_2, \ldots, x_n$ with the corresponding connections (weights) $w_1, w_2, \ldots, w_n$ being followed by a nonlinear (usually monotonically increasing) function that reads as follows

$$y = g(\mathbf{w}^T\mathbf{x} + \tau) = g(\sum_{i=1}^{n} w_i x_i + \tau)$$

(6)

where **w** is a vector of connections, τ is a constant term (bias) and “g” denotes some monotonically non-decreasing nonlinear mapping.

While some superficial and quite loose analogy between these processing units and logic neurons could be derived, one has to cognizant that these neurons do not come with any underlying logic fabric and hence cannot be easily and immediately interpreted.

Let us make two observations about the architectural and functional facets of the logic neurons we have introduced so far.

<u>incorporation of the bias term (bias) in the fuzzy logic neurons</u>

In analogy to the standard constructs of a generic neuron as presented above, we could also consider a bias term, denoted by $w_0 \in [0, 1]$, which enters the processing formula of the fuzzy neuron in the following manner

for the OR neuron

$$y = \mathop{S}_{i=1}^{n} (w_i t x_i) s w_0 \tag{7}$$

for the AND neuron

$$y = \mathop{T}_{i=1}^{n} (w_i s x_i) t w_0 \tag{8}$$

We can offer some useful interpretation of the bias by treating it as some nonzero initial truth value associated with the logic expression of the neuron. For the OR neuron it means that the output does not reach values lower than the assumed threshold. For the AND neuron equipped with some bias, we conclude that its output cannot exceed the value assumed by the bias. The question whether the bias is essential in the construct of the logic neurons cannot be fully answered in advance. Instead, we may include it into the structure of the neuron and carry out learning. Once its value has been obtained, its relevance could be established considering the specific value it has been produced during the learning. It may well be that the optimized value of the bias is close to zero for the OR neuron or close to one in the case of the AND neuron which indicates that it could be eliminated without exhibiting any substantial impact on the performance of the neuron.

<u>dealing with inhibitory character of input information</u> Owing to the monotonicity of the t-norms and t-conorms, the computing realized by the neurons exhibits an excitatory character. This means that higher values of the inputs (x_i) contribute to the increase in the values of the output of the neuron. The inhibitory nature of computing realized by "standard" neurons by using negative values of the connections or the inputs is not available here as the truth values (membership grades) in fuzzy sets are confined to the unit interval. The inhibitory nature of processing can be accomplished by considering the complement of the original input, that is $1\text{-}x_i$. Hence when the values of x_i increase, the associated values of the complement decrease and subsequently in this configuration we could effectively treat such an input as having an inhibitory nature.

2.3. An architectures of the fuzzy logic networks

The logic neurons can serve as building blocks of more comprehensive and functionally appealing architectures. The diversity of the topologies one can construct with the aid of the proposed neurons is surprisingly high. This architectural multiplicity is important from the application point of view as we can fully reflect the nature of the problem in a flexible manner. It is essential to capture the problem in a logic format and then set up the logic skeleton – a *conceptual* blueprint (by forming the and finally refine it parametrically through a thorough optimization of the connections. Throughout the entire development process we are positioned quite comfortably by monitoring the optimization of the network as well as interpreting its semantics.

The typical logic network that is at the center of logic processing originates from the two-valued logic and comes in the form of the fundamental Shannon theorem of decomposition of Boolean functions. Let us recall that any Boolean function $\{0,1\}^n \rightarrow \{0,1\}$ can be represented as a logic sum of its corresponding miniterms or a logic product of maxterms. By a minterm of "n" logic variables $x_1, x_2, \ldots, x_n$ we mean a logic product involving all these variables either in direct or complemented form. Having "n" variables we end up with 2^n minterms starting from the one involving all complemented variables and ending up at the logic product with all direct variables. Likewise by a maxterm we mean a logic sum of all variables or their complements. Now in virtue of the decomposition theorem, we note that the first representation scheme involves a two-layer network where the first layer consists of AND gates whose outputs are combined in a single OR gate. The converse topology occurs for the second decomposition mode: there is a single layer of OR gates followed by a single AND gate aggregating *or*-wise all partial results.

The proposed network (referred here as a logic processor) generalizes this concept as shown in Figure 4. The OR-AND mode of the logic processor comes with the two types of aggregative neurons being swapped between the layers. Here the first (hidden) layer is composed of the OR neuron and is followed by the output realized by means of the AND neuron.

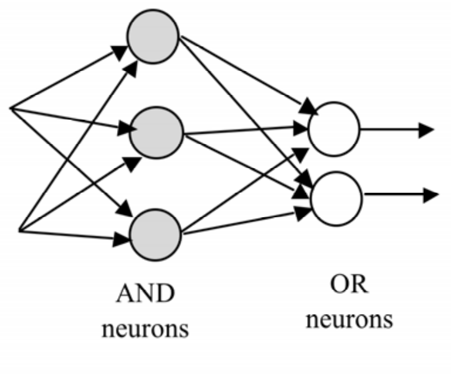


Figure 4. A topology of the logic processor (LP) in its AND-OR mode of realization

The logic neurons generalize digital gates by bringing essential learning capabilities and expanding the construct from its Boolean version to the multivalued alternative. The design of the network (viz. any fuzzy function) is realized through learning. If we confine ourselves to Boolean {0,1} values, the network's learning becomes an alternative to a standard digital design, especially a minimization of logic functions. The logic processor translates into a compound logic statement (for the time being we skip the connections of the neurons to emphasize the underlying logic content of the statement)

- if (input$_1$ *and... and* input$_n$) *or* (input$_d$ *and …and* input$_n$) then truth value of B_j

where the truth value of B_j can be also regarded as a level of "satisfaction" (activation) of the information granule B_j. Given the number of inputs and the number of outputs equal to "n" and "m", the logic processor generates a mapping from $[0,1]^n$ to $[0,1]^m$ thus forming a collection of "m" n-input fuzzy functions.

2.4. Interpretation aspects of the network

As the individual logic neurons of the logic processor are endowed with connections and each neuron has a well-specified nature of its logic processing, the network easily translates into a series of numerically quantified logic statements as illustrated above. To pursue the interpretation aspect in more detail, we look at the network shown in Figure 5 and given the values of the connections shown there we derive the quantified rule ("if-then" statement).

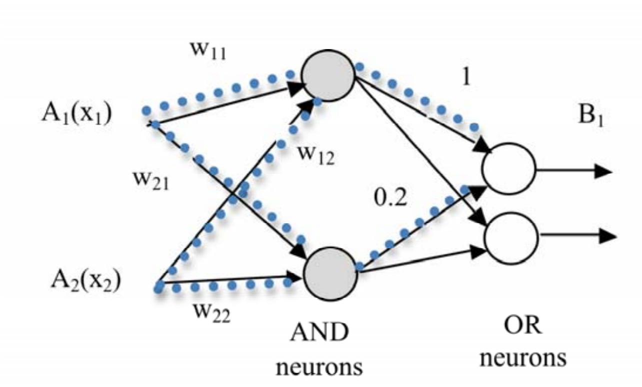


Figure 5. A two-input two-output fuzzy neural networks and their interpretation in the form of "if-then" statements; see a detailed description in the text.

The level of activation of the first output – information granule B_1 is formed by *or*-ing the outputs of the two AND neurons; we showed the individual contributions by a dotted line shown in the figure. The output of the first AND neuron translates into the following logic expression

$$[A_1(x_1) \textit{ or } w_{11}] \text{ and } [A_2(x_2) \textit{ or } w_{12}] \tag{9}$$

with the weights (connections) w_{11} and w_{12} quantifying an impact of the corresponding inputs x_1 and x_2 on the obtained output. For the second AND neuron we obtain

$$[A_1(x_1) \textit{ or } w_{21}] \text{ and } [A_2(x_2) \textit{ or } w_{22}] \tag{10}$$

These two partial results are aggregated *or*-wise by taking into account the connections of the OR neuron forming a single composite logic statement which can be viewed as the following rule

-if $\{[A_1(x_1)$ *or* $w_{11}]$ *and* $[A_2(x_2)$ *or* $w_{12}]$ *and* $1.0)\}$
or
$\{[A_1(x_1)$ *or* $w_{21}]$ *and* $[A_2(x_2)$ *or* $w_{22}]$ *and* $0.2\}$
then B_1

(11)

Noticeably, the contribution resulting from the first AND neuron is far more visible than the one produced by the second AND neuron.

This simple illustrative example demonstrates an important interpretability facet of fuzzy neural networks – one can immediately interpret (read) a collection of the rules just on a basis of the topology of the network. It becomes possible because of a logic character of processing delivered by the OR and AND neurons.

3. Interfacing with the real-world environment (data)

The inputs and outputs of the fuzzy neural network are positioned in the unit hypercube and as such exhibit some logic nature. Let us recall that, in general, the real-world data might have a very different nature. This means that a certain conceptual transformation to a more abstract level is required. One of the viable alternatives is to form information granules in the input and output spaces and perceive the data from the perspective established by such abstract viewpoint. Any original input and output are then regarded as degrees (levels) of activation of the information granules. These levels assume values in the [0,1] interval. The available data being to design the model are provided in the form of the input-output pairs $(\mathbf{x}_k, f_k)$, k=1, 2, …, N.

There are somewhat different ways of the design of information granules in the input and output spaces:
Information granules in the input space More specifically, in the input space (which typically comprises a number of input variables), we form information granules by forming fuzzy clusters. The method of Fuzzy C-Means [6][13] is a commonly used alternative. The underlying essence is to cluster the data, that is reveal a structure in the input data forming a collection of "c" prototypes $\mathbf{v}_1$, $\mathbf{v}_2$, .., $\mathbf{v}_c$ and the corresponding partition matrix U describing degrees of membership of each input data $\mathbf{z}_k$ to the corresponding cluster (which are contained in the [0,1] interval).
Information granules in the output space Again a before we run the FCM method, assuming a certain number of clusters "m" on the one-dimensional output data $\{f_k\}$, k=1, 2, …, N and form the prototypes. To enhance interpretability we can also construct

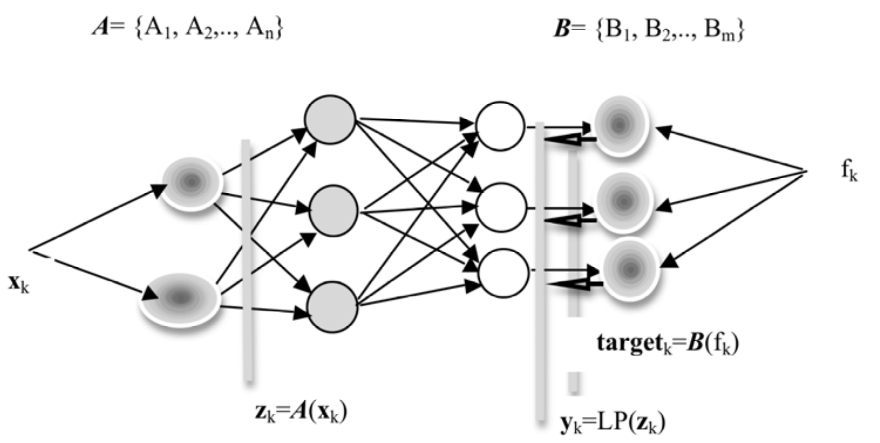
triangular membership functions based on the results of clustering. More specifically, the prototypes form the modal values of these fuzzy sets and the adjacent fuzzy sets overlap at the level ½. This family of fuzzy sets realizes a *lossless* granulation-degranulation effect [15] meaning that there is a mapping from **R** to the membership degrees $u_1, u_2, \ldots, u_m$ (granulation phase) and a subsequent weighted scheme (degranulation phase) using which we "reconstruct" the original data without any reconstruction (granulation-degranulation) error. The zero value of the error is achieved when using a family of triangular fuzzy sets with the ½ overlap between the adjacent fuzzy sets. These properties of the fuzzy sets occur in presence of the fuzzy sets constructed as presented above.

When taking a view at the interfacing mechanism and its realization, we distinguish between a learning model (when the logic processor is constructed) and the usage mode when the logic processor generates an output for a given input.

In the learning mode, Figure 6, we are provided with the training data ($\mathbf{x}_k$, f_k) to be used to construct the logic processor. We assume that the collections of information granules for the input and output spaces are given. Denote them by $\boldsymbol{A}= \{A_1, A_2, \ldots, A_n\}$ and $\boldsymbol{B} = \{B_1, B_2, \ldots, B_m\}$. The data are transformed nonlinearly – expressed via the information granules giving rise to the following transformed results that subsequently are used for the training of the logic processor. More formally, we obtain

$$\begin{gathered}\mathbf{x}_k \rightarrow \boldsymbol{A} \rightarrow \mathbf{z}_k \in [0,1]^n \\ f_k \rightarrow \boldsymbol{B} \rightarrow \mathbf{target}_k \in [0,1]^m\end{gathered} \tag{12}$$

Note that the transformation results in the elements located in the unit hypercubes. In this sense, the above transformation produces a required normalization of the data, which could be beneficial from the learning perspective.



$\boldsymbol{A}= \{A_1, A_2,.., A_n\}$
$\boldsymbol{B}= \{B_1, B_2,.., B_m\}$
f_k
$\mathbf{x}_k$
$\mathbf{target}_k=\boldsymbol{B}(f_k)$
$\mathbf{z}_k=\boldsymbol{A}(\mathbf{x}_k)$
$\mathbf{y}_k=LP(\mathbf{z}_k)$

Figure 6. Interfaces of logic processor- the design (learning) mode; shown is a way of forming internal representations of $\mathbf{x}_k$ and f_k located in the unit hypercubes

In the usage mode, the input x is transformed by the elements of A producing a vector in $[0,1]^n$. This, in turn, is processed by the logic processor forming a certain output vector **y** in the $[0,1]^m$ hypercube. To return a result in an "external" format, it us further transformed (decoded) with the use of the elements of B. Typically, one considers here an aggregation of the activation levels of B_j's along with their numeric representative such as e.g., modal values, cf. [15]

4. The design of fuzzy logic network

The design of the network comprises two main phases; (a) the development of information granules, and (b) optimization (learning) of the connections of the network. The first phase is very much about a way of interfacing the logic processing realized by the logic neurons with the real-world data. As noted in the previous section, the interfacing is formed through a series of information granules built in the input and output space. Typically such information granules are constructed through clustering or fuzzy clustering. One of the common ways of building fuzzy sets in the input and output space is to apply Fuzzy C-Means (FCM) [6] As a result, we obtain a collection of the prototypes in the input and output spaces, which, in the sequel give rise to membership function of A_i and B_j. For instance the membership function of A_i reads as follows

$$A_i(\mathbf{x}) = \frac{1}{\sum_{j=1}^{n}\left(\frac{\|\mathbf{x}-\mathbf{v}_i\|}{\|\mathbf{x}-\mathbf{v}_j\|}\right)^{2/(p-1)}}$$

(13)

where $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c$ are the prototypes of the clusters. The parameter "p" is a so-called fuzzification coefficient assuming values greater than 1, $p > 1$. Its value affects the shapes of the resulting membership functions of A_i. The typical value of the fuzzification coefficient encountered in the literature is equal to 2.

There are several crucial parameters of the network one has to decide upon before engaging in the parametric learning, viz. the optimization of the values of the connections. They include the number of AND nodes in the hidden layer (h) and a choice of a certain t-norm and t-conorm (note that they need not to be dual). The parametric learning involves the adjustment of the connections of the neurons and is quite standard. If we consider a training data set in the for of the pairs $\mathbf{D} = \{(\mathbf{x}_k, \mathbf{target}_k)\}$, k=1, 2, …, N and a sum of squared errors,

$$Q = \sum_{k=1}^{N}(\mathbf{target}_k - N(\mathbf{x}_k))^T(\mathbf{target}_k - N(\mathbf{x}_k))$$

(14)

where $N(\mathbf{x}_k, \mathbf{connections})$ is a vector of outputs of the fuzzy network obtained for the given input $\mathbf{x}_k$, the learning follows the update iterative scheme

$$\mathbf{connections}(\text{iter}+1) = \mathbf{connections}(\text{iter}) - \beta \nabla_{\mathbf{connections}(\text{iter})} Q$$

(15)

5. Granularity of information as a design asset and its optimal allocation

As highlighted earlier, the crux of the principle of information granularity is to allocate information granularity to the fuzzy logic network. More specifically, the numeric values of the connections are generalized (abstracted) to the granular connections in the form of some intervals being included in the unit interval. The emergence of the granular (interval) connections is legitimate. What we are proposing here stresses a role of information granularity being viewed as an important design asset, which needs to have prudently exploited. A way of building intervals around the original numeric values of the connections can be referred to as an *optimal* granularity allocation. In a nutshell, the formation of granular (interval) connections should result in the optimization of a certain performance index being reflective of the quality of granular fuzzy neural networks.

In what follows, we start with a detailed discussion on the evaluation of the quality of these networks. We show that the quality of such network can be optimized through a suitable allocation of available information granularity. Let us also note that given a certain level of information granularity assuming a certain value α assuming values in the unit interval, it associates with the given numeric connection "w" by forming an interval of length a distributed around z with eventual clipping of the range (if required). This means that in the simplest scenario the granulation of w, $\boldsymbol{G}(\mathrm{w})$ results in an interval $[\max(0, w-\alpha/2), \min(1, w+\alpha/2)]$.

The essence of the granulation of the fuzzy logic network is visualized in Figure 7.

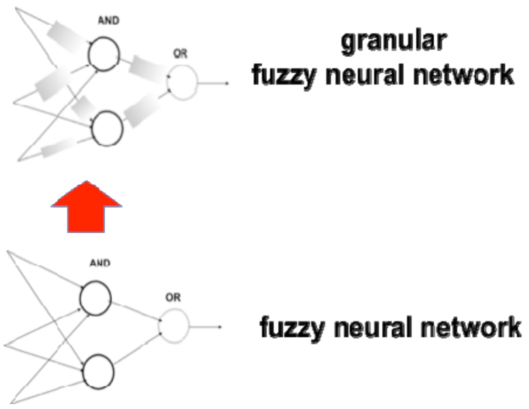


Figure 7. From fuzzy neural network to its granular abstraction (generalization); small rectangular shapes emphasize the interval-valued character of the granular connections

As the connections of the logic neurons are now granular (represented in the form of intervals), the output of network becomes granular as well. To emphasize that, let us look at the OR neuron described by (7) where the connections are made granular, that is $G(w_{ij}) = [w_{ij-}, w_{ij+}]$. We have the following expression

$$Y_i = [y_{i-}, y_{i+}] = \overset{n}{\underset{j=1}{S}}(G(w_{ij})\ t\ u_j)$$

(16)

which, in virtue of the monotonicity of t-norms and t-conorms, results in the bounds of Y_i to be equal to

$$Yi = [\overset{n}{\underset{j=1}{S}}(G(w_{ij-})\ t\ u_j), \overset{n}{\underset{j=1}{S}}(G(w_{ij+})\ t\ u_j)]$$

(17)

The quality of the granular fuzzy neural network, assuming that a certain level of granularity α has resulted in the corresponding granular connections, can be evaluated in several ways.

Intuitively, as the connections are granular (interval-valued), the output produced by the network is also of interval-valued nature. Ideally, one would anticipate that the outputs of the granular network should include the original data **D'**. Consider $\mathbf{x}_k \in \mathbf{D}'$ with the cardinality of **D'** equal to N'. After the transformation of $\mathbf{x}_k$ by the elements of ***A*** yields the vector $\mathbf{z}_k \in [0,1]^n$. This means that each of the outputs of the granular neural network comes in the form of the interval $Y_{kj} = [y_{j-}, y_{j+}] = G(N(\mathbf{z}_k))_j$, j=1, 2,…, m. Furthermore through the transformation realized by the elements of ***B***, f_k translates into $\mathbf{target}_k = \boldsymbol{B}(f_k) \in [0,1]^m$. The quality of the granular network can be assessed by counting how many times the inclusion relationship $\text{target}_{kj} \in G(N(\mathbf{x}_k))_j$ holds. In other words, the performance index is expressed in the following form

$$\kappa = \frac{\sum_{j=1}^{m}\sum_{k=1}^{N'}\{\text{card}((k,j) \mid \text{target}_{kj} \in G(N(\mathbf{z}_k)_j\}}{N'*m}$$

(18)

Ideally, we could expect that this ratio is equal to 1. Of course κ becomes a nondecreasing function of α , κ(α) so less specific information granules (higher values of α) produce better coverage of the data but at an expense of the obtained results being less specific. Note also that the values of κ depend upon the predetermined level of α, emphasized here by the notation κ(α). Here a monotonicity property is satisfied, namely κ(α) is a nondecreasing function. Higher values of α imply higher values of coverage of the fuzzy sets of conclusion. To achieve a global assessment of the quality of the granular fuzzy rules, we integrate or do a summation (in case of discrete values of α) of the values of κ(α), which results in a index κ independent from the assumed level of granularity, κ = $\int_0^1 \kappa(\alpha)d\alpha$. Note that it is nothing but an area under the curve, AUC. The plot κ(α) itself could be helpful in a visual inspection of increases of the coverage versus the increased values of α. Some example plots of this relationship are shown in Figure 8.

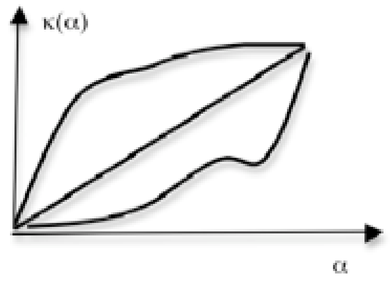


Figure 8. Example plots of κ(α): (a) uniform increase of coverage, (b) increase of coverage exhibiting a visible jump present at some low values of α, (c) increased coverage level occurring at higher values of α.

Along with the coverage criterion, we can look at the quality of the information granule of the output formed by the granular logic network, that is a length L of the interval L(G(N($\mathbf{x}_k$)) or its average value,

$$L = \frac{1}{M}\sum_{k=1}^{M} L(G(N(\mathbf{x}_k)) \tag{19}$$

Note that the criteria (18) and (19) are in conflict: while high values of (18) are preferred, lower values of (19) are advisable.

In what we follows, we discuss some more advanced ways of allocating information granularity to the individual connections of the network (not all connections need to be granulated to the same extent), so that the performance indices (18) and (19) can be optimized (maximized and minimized, respectively) or their aggregate could be optimized.

5.1. Protocols of allocation of information granularity

An allocation of the available information granularity to the individual connections of the network can be realized in several different ways depending how much diversity one would like to exploit in the allocation process. Here, we discuss several protocols of allocation of information granularity, refer also to Figure 9:
P_1: uniform allocation of information granularity. This process is the simplest one and in essence does not call for any optimization. All weights (connections) are treated in the same way and become replaced by the same interval (the length of the interval is the sane for all connections).
P_2: uniform allocation of information granularity with asymmetric position of intervals around the original connections of the network.
P_3: non-uniform allocation of information granularity with symmetrically distributed intervals of information granules.
P_4: non-uniform allocation of information granularity with asymmetrically distributed intervals of information granules.

P_5: An interesting point of reference, which is helpful in assessing a relative performance of the above methods, is to consider a random allocation of granularity. By doing this, one can quantify how the optimized and carefully thought out process of granularity allocation is superior over a purely random allocation process.
In all these protocols, we assure that the allocated information granularity meets the constraint of the total granularity that is αH where H stands for a number of all the connections of the network.

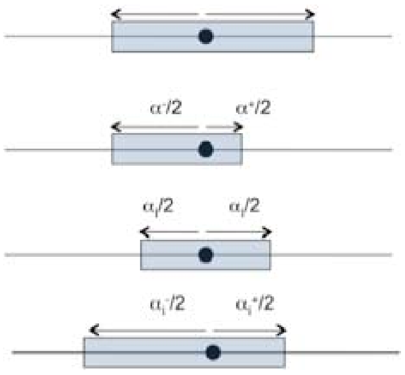


Figure 9. Protocols of allocation of information granularity and the resulting granular realization of the fuzzy sets of condition

No matter whether we are considering swarm optimization or evolutionary techniques (say, genetic algorithms), the respective protocols call for a certain content of the particle or a chromosome. The length of the corresponding string depends upon the protocol, which becomes longer with the increased specialization of granularity allocation.
Having considered all components that in essence constitute the environment of allocation of information granularity, we can bring them together to articulate a formal optimization process.
Assume that a certain fuzzy neural network has been provided. Given a certain protocol of allocation of information granularity P, determine such allocation **I**, call it $\mathbf{I}_{opt}$, so that the value of the coverage index κ becomes maximized,

$$\mathrm{Max}_{\mathbf{I}}\ \kappa \tag{20}$$

The expression (xx) leads to a combinatorial optimization problem and as such requires more advanced optimization techniques. Here the methods of evolutionary optimization or swarm optimization could be viable alternatives to consider. Let us also remark that with the use of $\mathbf{I}_{opt}$ one can assess the effectives of various strategies (protocols) of allocation of information granularity.
In light of the nature of the two optimization tasks present here, namely (a) a selection of the optimal subset of the rules, and (b) the maximization of the coverage, these two can be handled by the corresponding nested optimization processes of evolutionary optimization. In other words, for a subset of the rules generated by the optimization process at the upper level, one carries out the optimal allocation of information granularity following a certain format of the assumed protocol.

Alluding to the refinement of the protocols of allocation of information granularity, we conclude the following relationship among I_{opt} resulting from the respective protocols is satisfied,

$$\mathbf{I}_{opt}(P_5) \le \mathbf{I}_{opt}(P_1) \le \mathbf{I}_{opt}(P_2) \le \mathbf{I}_{opt}(P_3) \le \mathbf{I}_{opt}(P_4) \tag{21}$$

The corresponding search spaces associated with the realization of the protocols (with the nested property given by (xx)) start exhibiting higher dimensionality. Obviously, the numeric values of this performance index are not known in advance and it is of interest to compare them and thus quantify how particular protocols are effective for a given fuzzy logic network.

6. Experimental studies

We report a series of experimental results, which help quantifying the performance of the proposed approach. In particular, we contrast the advantages, which are brought by an effective allocation of available information granularity. The achieved improvement is visualized through the coverage-granularity curves and reported in terms of the corresponding values of the AUC.

Synthetic data This single input – single output data consists of 10 data points using which a fuzzy logic network has been constructed, Figure 10 (a). The knowledge transfer is realized in presence of 5 data points shown in Figure 10 (b). One can note that in spite some similarity, the data exhibit some difference in comparison with the trend visible in the original data.

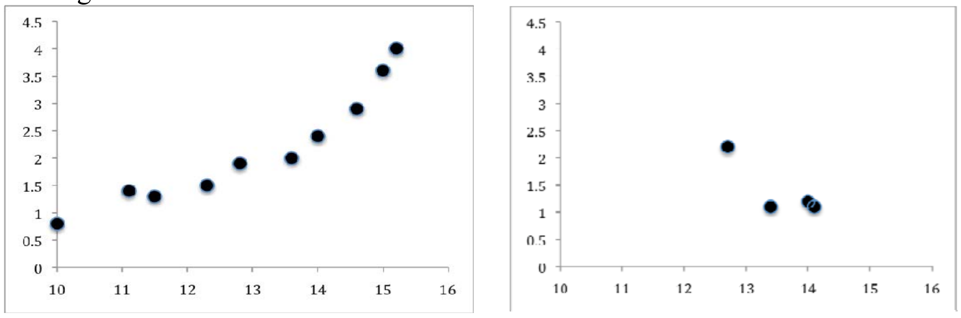
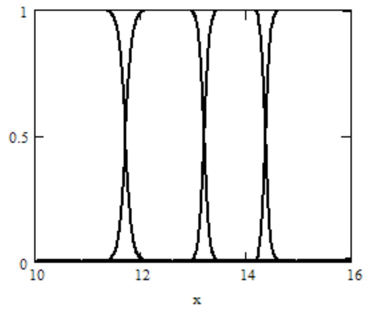


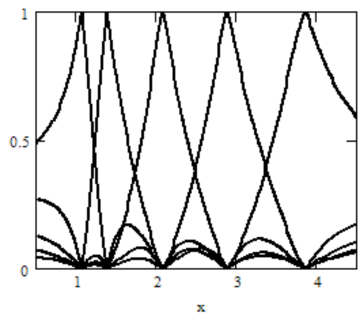
Figure 10. Data sets: (a) use in the learning of the fuzzy logic network, and (b) in knowledge transfer

The granulation of data (as discussed in Section xx) has been realized by running the FCM algorithm with the following values of its parameters: input : c =4, p= 1.3, output: c =5 p =2.5 (the values of these parameters were selected for illustrative purposes; in particular we show how the fuzzification coefficient impacts the shape of the corresponding membership functions).The obtained prototypes are as follows:

input space: 14.93 10.85 13.82 12.59
output space: 2.88 1.39 1.07 2.08 3.85

The plots of the membership functions in the input and output spaces are presented in Figure 11.





(a) (b)

Figure 11. Membership functions in the input (a) and output (b) space

We start with the two schemes of granularity allocation where no optimization has been invoked, that is (a) uniform allocation, symmetric case, and (b) uniform allocation, asymmetric case. For reference, we also run a scheme where the granularity allocation has been done randomly (by drawing values of granularity from the uniform distribution spanned over the unit interval). The results are reported in terms of the performance index regarded as a function of α; see Figure 12.

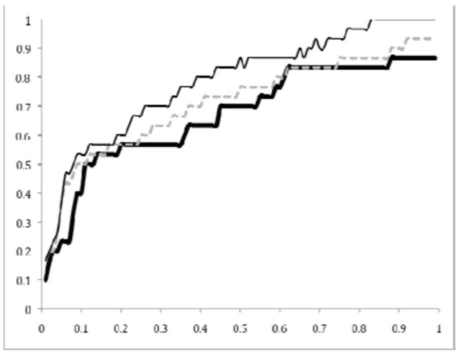


Figure 12. Performance index versus the values of α: (a) (b) , and (c)

As could have been anticipated, the uniform allocation with a symmetric granulation of the connections of the network is the weakest strategy. Some improvement is noted when we allow for asymmetric location of the interval connections. The random mechanism of granularity allocation yields some improvement as the corresponding curve is elevated in comparison with the two others.

The overall performance is quantified by looking at the area under the corresponding curves; here we obtain

AUC = 0.663 for (a) AUC = 0.707 for (b) AUC = 0.778 for (c) which demonstrates that the random allocation of the granularity exhibits superiority over the systematic albeit very rigid strategy of granularity allocation. The results shown in Figure xx and quantified in terms of the AUC values

AUC = 0.776 for (a) AUC = 0.857 for (b)

clearly show that the PSO strategy produces better results in case of asymmetrically allocated information granularity. Notably the symmetric information granules of the connections are not that advantageous; the PSO led to the higher values of the AUC (when compared to the uniform distribution of granularity), it did not manage to produce better results as those obtained by the random allocation mechanism. One has stress, however, that when running the random allocation, there was no restriction as to the symmetry of the granular connections.

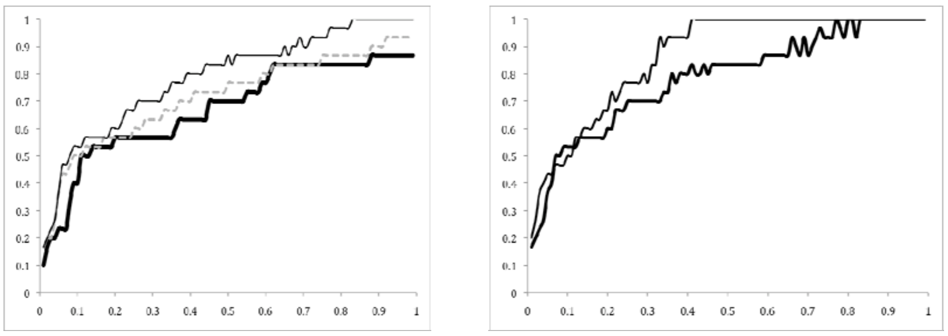


Figure 13. Results of the PSO -performance index versus α: (a) symmetric construction of the connections, (b) asymmetric granular connections

Software data The data come from the study carried by Miyazaki et al. [12] (see also http://promisedata.org/?cat=6) and is concerned with modeling of software cost estimation, namely man-months regarded as a function of several descriptors of the software project including the number of lines of code (KLOC), number of screens, number of forms, files, etc. The data set collected by Fujitsu Large Systems Users Group coming from 48 systems in 20 companies. This speaks loudly to the diversity of the data. The data set consists of 47 data points (one which deals with an extremely large project has not been included in this study). The five data points (which concerns a different system) are used for the realization of knowledge transfer. As described, running the FCM results in a number of information granules formed in the input and output space. The number of clusters in the input and output space is equal to 4 and 5, respectively while the values of the fuzzification coefficient are the same as in the previous experiment. The obtained prototypes are as follows

input space:

$\mathbf{v}_1$= [27.29 17.23 8.64 12.06 246.66 178.48 438.84]^T

$\mathbf{v}_2$= [51.72 20.28 19.48 37.68 324.17 404.30 1162.24]^T

$\mathbf{v}_3$=[284.08 114.87 47.69 69.92 1755.56 876.98 2444.10]^T

$\mathbf{v}_4$= [90.34 39.73 43.48 29.77 844.07 1040.83 1123.52]^T

output space: 11.69   37.08 338.76 167.29   61.78

Each of these information granules comes with a well-defined semantics. For instance, the prototypes in the output space can be interpreted as follows:

low (11.69) …*medium* (61.78)… *large* (338.76)

Running the same set of experiments as before, the main results are succinctly reported in Figure 14.

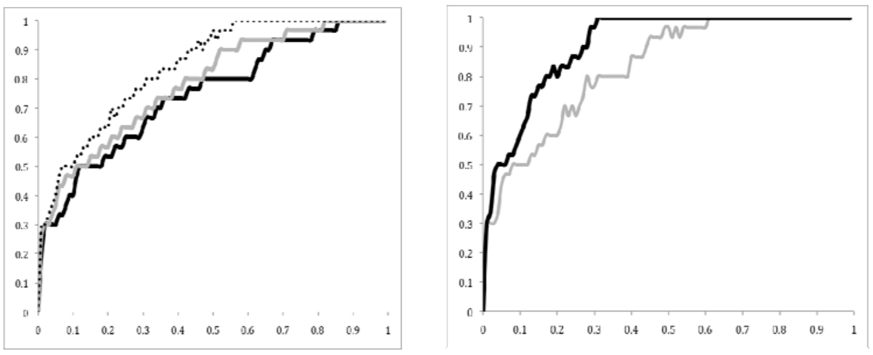


Figure 14. Results of the PSO -performance index versus α: (a) symmetric construction of the connections, (b) asymmetric granular connections

They lead to some general conclusions regarding the effectiveness of the different strategies of granularity allocation. The two ones without any optimization lead to the weakest results quantified by the values of AUC being equal to 0.741 (symmetric granular connections) and 0.778 (asymmetric granular connections). The random allocation exhibits some competitive edge, AUC = 0.835 however it is surpassed by the PSO solution (which is indicative of the effectiveness of the optimization method). More specifically, the quality of solutions is characterized by the following values of the AUC: 0.827 for symmetric granular connections, and AUC = 0.902 for asymmetric ones.

7. Conclusions

The emergence of granular models is a direct consequence of knowledge transfer with the mechanism of information granularity used to quantify a level of abstraction brought to the original model. The process of granularity allocation (coming with several detailed protocols) is optimized by means of the PSO.

Different models of granular connections of fuzzy neural networks can be envisioned. While here we have considered interval-valued connections as being quite appealing and computationally feasible, it is worth stressing that the underlying concept is general enough and any other formalism of information granules could be considered as well. For instance, the granular augmentation of fuzzy neural networks can be specified as

(a) fuzzy fuzzy (fuzzy^2) neural networks where the connections are described by fuzzy sets

(b) rough fuzzy neural networks in which the connections are modeled as rough sets

There is also an interesting facet of granularity of input data of the model. Those themselves can be viewed as information granules (meaning that the available inputs are approximate only being a result of estimation rather than a detailed measurement). This source of granularity could be easily incorporated into the granular model. Subsequently one could study its impact on the propagation of granularity and its manifestation through a level of granularity of the results produced by the granular model.

References

1. J. Aroba, J. J. Cuadrado-Gallego, M-A. Sicilia, I. Ramos, E. García-Barriocanal, Segmented software cost estimation models based on fuzzy clustering, ***J. of Systems and Software***, 81, 11, 2008, 1944-1950.
2. M. Azzeh, D. Neagu, P. I. Cowling, Analogy-based software effort estimation using fuzzy numbers, ***J. of Systems and Software***, 84, 2, 2011, 270-284.
3. A.Bargiela, W. Pedrycz, ***Granular Computing: An Introduction***, Kluwer Academic Publishers, Dordrecht, 2003.
4. A. Bargiela, W. Pedrycz, Granular mappings, ***IEEE Transactions on Systems, Man, and Cybernetics-part A***, 35, 2, 2005, 292-297.
5. A. Bargiela, W. Pedrycz, Toward a theory of Granular Computing for human-centered information processing, ***IEEE Transactions on Fuzzy Systems***, 16, 2, 2008, 320 – 330.
6. J.C. Bezdek, ***Pattern Recognition with Fuzzy Objective Function Algorithms***, Plenum Press, N. York, 1981.
7. A.F. Gobi, W. Pedrycz. Fuzzy modeling through logic optimization, ***Int. J. of Approximate Reasoning***, 45, 2007, 488-510.
8. L. M. Karg, M. Grottke, A. Beckhaus, A systematic literature review of software quality cost research***, J. of Systems and Software***, 84, 3, 2011, 415-427.
9. Y-F. Li, M. Xie, T.-N. Goh, Adaptive ridge regression system for software cost estimating on multi-collinear datasets, ***J. of Systems and Software***, 83, 11, 2010, 2332-2343.
10. C. López-Martín, C. Yáñez-Márquez, A. Gutiérrez-Tornés, Predictive accuracy comparison of fuzzy models for software development effort of small programs, ***J. of Systems and Software***, 81, 6, 2008, 949-960
11. N. Mittas, L. Angelis, Visual comparison of software cost estimation models by regression error characteristic analysis, ***J. of Systems and Software***, 83, 4, 2010, 621-637.
12. Y. Miyazaki, M. Terakado, K. Ozaki, Robust regression for developing software estimation models, J. of Systems and Software, 1994, 27, 3-16.
13. W. Pedrycz, ***Knowledge-Based Clustering: From Data to Information Granules***, J. Wiley, Hoboken, NJ, 2005.
14. W. Pedrycz, F. Gomide, ***Fuzzy Systems Engineering: Toward Human-Centric Computing,*** John Wiley, Hoboken, NJ, 2007

15. W. Pedrycz, J. Valente de Oliveira, A development of fuzzy encoding and decoding through fuzzy clustering, ***IEEE Transactions on Instrumentation and Measurement***, 57, 4, 2008, 829 – 837.
16. w. Pedrycz, P. Ekel, R. Parreiras, ***Fuzzy Multicriteria Decision-Making: Models, Methods and Applications***, J. Wiley, J. Wiley, Chichester, 2011.
17. F. Reyes, N. Cerpa, A. Candia-Véjar, M. Bardeen, The optimization of success probability for software projects using genetic algorithms, ***J. of Systems and Software***, in press, 2011
18. L.A. Zadeh, Towards a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic, ***Fuzzy Sets and Systems***, 90, 1997, 111-117.