Enhanced Fuzzy-Relational Neural Network with Alternative Relational Products

Efraín Mendoza-Castañeda[1], Carlos A. Reyes-García[1], Hugo Jair Escalante[1], Wilfrido Moreno[2], and Alejandro Rosales-Pérez[1]

[1] Instituto Nacional de Astrofísica, Óptica y Electrónica, Tonantzintla, Puebla, Mexico
[2] University of South Florida Tampla, Florida, USA
efrain.mendoza.c@inaoep.mx

Abstract. This paper describes an extension of fuzzy relational neural networks (FRNNs) that aims at improving their classification performance. We consider Pedrycz's FRNN, which is one of the most effective and popular models. This model has traditionally used a single relational product (Circlet). The extension described in this paper consists in allowing applying other relational products in the training phase to the basic FRNN, looking to increase its predictive capabilities. The relational products considered for the extension are the so called BK-Products: SubTriangle, SupTriangle and Square; in addition, we propose the use of more general operators (t-norms and s-norms) in their definitions, which are also applied to the Circlet relational product. We explore the effectiveness of this extension in classification problems, through testing experiments on benchmark data sets with and without noise. Experimental results reveal that the proposed extension improves the classification performance of the basic FRNN, particularly in noisy data sets.

Keywords: Fuzzy Relational Neural Networks, Neuro-Fuzzy Systems, Relational Products.

1 Introduction

Neural and fuzzy systems combine naturally to resemble an adaptive system with sensory and cognitive components [12]; these methods are known as neuro-fuzzy hybrid systems and can be grouped into [10]: (I) Fuzzy Neural Networks, where the network is capable to process fuzzy information; and (II) Neuro-Fuzzy systems involving a Fuzzy Inference System combined with a neural network which provide learning ability. This paper focuses on the first approach, where neurons perform operations from fuzzy sets theory instead of the common arithmetic operations, concretely we focus on Fuzzy Relational Neural Networks (FRNNs).

In FRNNs the weights are replaced by fuzzy relations, in this way, FRNNs account for the uncertainty that may exist in training data. Several FRNNs have been proposed so far, being the model from Pedrycz the most known one [11]. However, this model is restricted to a single relational product (Circlet).

E. Bayro-Corrochano and E. Hancock (Eds.): CIARP 2014, LNCS 8827, pp. 666–673, 2014.

We propose an extension to Pedrycz's FRNN to incorporate alternative relational products that have not been considered yet for the training phase. Specifically, we consider *SubTriangle*, *SupTriangle* and *Square* relational products, additionally we consider more general operators (i.e., *t-norms*, *s-norms*) in their definitions. We adapt the learning algorithm (a variant of the backpropagation algorithm) described in [11] for training FRNNs with the alternative products. We performed experiments with the extended FRNNs in benchmark data with and without noise. Results reveal that FRNNs trained with different relational products can improve the classification performance than the base model.

The rest of the paper is organized as follows. Section 2 introduces the basic FRNN. Section 3 reviews related work. Section 4 introduces the proposed extension. Section 5 reports experimental results. Finally, Section 6 presents conclusions and future work directions.

2 Fuzzy Relational Neural Networks

A feed-forward single-layer perceptron consists of a collection of input nodes $X = \{x_1, ..., x_m\}$, a set of output nodes $Y = \{y_1, ..., y_l\}$, and a weights matrix $W = \{w_{ij} | i \in m, j \in l\}$, where l are the classes to which input patterns can belong to. The output of node y_j is given by $y_j = f\left(\sum_i x_i w_{ij}\right)$.

Pedrycz [11] replaces weights by fuzzy relations. The perceptron's weight matrix is replaced by a fuzzy relational matrix which represents a fuzzy relation from X to Y, $R = \{xRy | x \in X, y \in Y\}$, so that the connection between x_i and y_j has a relational value $R(x_i, y_j)$. The output values Y are generated by the relational product (RP) of the inputs set X and the relations R, i.e. $Y = X \circ R$. For this network, Pedrycz uses the circlet product defined by the *max-min* composition [15]. This composition is represented by $y_j = max\left(min\left(x_i, R\left(x_i, y_j\right)\right)\right)$. In [11], it is also proposed a fuzzy equality index that is based in the Łukasiewicz implication, this index provides a way to evaluate the model's performance during the training phase. This measure is used with gradient descent techniques to update the weights of the matrix in a similar fashion to the standard backpropagation (BP) algorithm. Since gradient descent techniques require the objective function to be derivable, Pedrycz also proposes an approximate of the derivative for the *max-min* composition.

Reyes-García proposed in [12] a framework to take advantage of Pedrycz's FRNN, with the particularity that this FRNN uses different relational products in the *processing phase* (described shortly). In this architecture the input layer is formed by $N \times n$ neurons, each of which corresponds to one of the N linguistic terms assigned to each of the n inputs. The output layer is composed by l neurons, each of which belongs to one of l classes. There exists a connection between each node in the input layer to every node in the output layer. Figure 1 shows the framework proposed. The FRNN operation is divided in two phases: (1) the learning phase is responsible for directing the adjustment of the values of the relations matrix: (2) the processing phase is in charge of calculating the membership degree to the classes of each input pattern; in this point the outputs are

fuzzy values, here the Decision Making Module (DMM) interprets the outputs depending on the application, e.g., for classification tasks the DMM assigns the label with the highest membership degree to the input pattern.

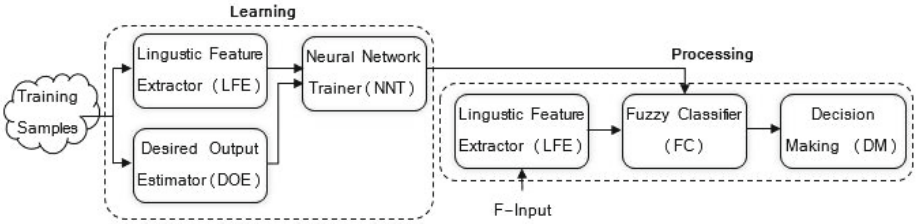

Fig. 1. Block diagram of the framework of the system based FRNN from [12]

3 Related Work

The *max-min* composition is one of the most frequently used for training FRNNs, see e.g., [5,6,7,11,12,13]. W. Pedrycz proposed the basic structure of the FRNN [11] and Blanco et al. integrated the concept of "soft derivation" to approximate the actual derivative [5]. Valente de Oliveira [7] expanded the composition to *max-t*. Whereas Reyes-García [12] extended the network developed by Pedrycz[11] to incorporate the *BK* relational products in the processing phase [2]-[8]. Davis & Kohout [6] incorporated generalizations of the BK RPs. In [4] Barajas & Reyes develop a genetic algorithm (to select the number of linguistic terms, type of membership functions, and learning rate); with the same approach Rosales et al. [14] select, in addition to the aforementioned parameters, the values for the membership functions. Other works (see e.g., [1]) focus on the use of derivable norms. From the above reviewed works we note that most methods use the *max-min* composition for training FRNNs.

4 Incorporating Alternative RPs for Training FRNNs

This paper proposes to use different RPs in the training phase of FRNNs, as well as the use of *t-norms* and *s-norms* in the RP's definitions. Our work is inspired in part by the observations of Bandler & Kohout in [3], related to the use of particular implication operators adequate to specific applications. It follows that if we only trains an FRNN with a specific composition (*max-min* in this case), its ability to represent other relationships between structures is limited. Therefore, it is necessary to provide the FRNN with the ability of replacing the training RP by another more suitable to the problem under analysis. In the rest of this section we describe the way in which alternative RPs can be incorporated in the training phase of FRNNs.

4.1 Training Phase

Switching from one RP to another for training an FRNN is not a trivial task and it cannot be done in a transparent way. This is mainly due to the fact that the learning algorithm (BP) used by the FRNN is a method based on gradient descent and therefore requires derivable operators.

Pedrycz incorporates an equality index Q expressed by fuzzy implications [11], this index serves as a measure to evaluate the network's error, when calculating the derivative of this error with respect to the weights, we obtains the following:

$$\Delta w = \frac{\partial Q}{\partial w} = -\sum_{i:y_i>t_i} \frac{\partial f(x_i; w, \vartheta)}{\partial w} + \sum_{i:y_i<t_i} \frac{\partial f(x_i; w, \vartheta)}{\partial w} \tag{1}$$

where x_i is the input pattern, w is the relation matrix and ϑ is the bias vector. The rest of the derivative (Δw) can be calculated when the function f is specified. For example, if we take into account the definition of the *circlet* RP ($\circ$) with a bias, $y_k = \vee\left(\vee\left(x_{ij} \wedge w_{kj}\right), \vartheta_k\right)$ where $k = 1, .., l$, we have (the indexes have been omitted for clarity):

$$\frac{\partial(\vee(w \wedge x) \vee \vartheta)}{\partial w} = \frac{\partial(\vee(w \wedge x) \vee \vartheta)}{\partial \vee (w \wedge x)} * \frac{\partial(\vee(w \wedge x))}{\partial(w \wedge x)} * \frac{\partial(w \wedge x)}{\partial w} \tag{2}$$

this derivative is given in terms of *t-norms* and *s-norms*, these operators have to be replaced by implementations that comply with the restrictions inherent to these. Thus if we replace the *t-norms* by the *min* operator and the *s-norms* by the *max* operator (considering that we know the approximation of their derivatives [11,5]), the derivative stays as follows:

$$\frac{\partial(\vee(w \wedge x) \vee \vartheta)}{\partial \vee (w \wedge x)} = \begin{cases} 1 & \vee(w \wedge x) \geq \vartheta, \\ 0 & otherwise. \end{cases} \qquad \frac{\partial(w \wedge x)}{\partial w} = \begin{cases} 1 & a \leq x, \\ 0 & otherwise. \end{cases}$$

$$\frac{\partial(\vee(w \wedge x))}{\partial(w \wedge x)} = \begin{cases} 1 & (w \wedge x) \geq (\vee(w \wedge x), \\ 0 & otherwise. \end{cases}$$

as can be seen, the $max - min$ derivative is reduced to a number of cases, which are identical to those shown by Pedrycz [11]. By adopting this approach we can add more RPs for training, we just need to define the derivative of the RP to the point shown in Equation (2) and the rest of the process is transparent. In this work, the RPs that are incorporated to the training phase of FRNNs are the BK products (by Bandler-Kohout): SupTriangle, SubTriangle and Square; this is due to their proved quality to represent relationships between structures [9]. We provide the derivative and definitions of each of these products in Table 1, these definitions are one of the main contributions of this work. To complete the training process we need to define the *t-norms*, *s-norms* and implications, with their respective derivatives, that will be used by the RPs. The operators were chosen for their wide use in the literature, we provide the details of these in Table 2 (the operators used for training are those in which the derivative has been defined).

Table 1. RPs to be incorporated in the training phase of FRNNs

Rp	Definition	Derivative
Circlet($\circ$)	$\vee_j(R_{ij} \wedge S_{ij})$	$\frac{\partial(\vee(w\wedge x)\vee\vartheta)}{\partial\vee(w\wedge x)} * \frac{\partial(\vee(w\wedge x))}{\partial(w\wedge x)} * \frac{\partial(w\wedge x)}{\partial w}$
SubTriangle($\triangleright$)	$\wedge_j(R_{ij} \rightarrow S_{jk})$	$\frac{\partial(\wedge(x\rightarrow w)\vee\vartheta)}{\partial(\wedge(x\rightarrow w))} * \frac{\partial(\wedge(x\rightarrow w))}{\partial(x\rightarrow w)} * \frac{\partial(x\rightarrow w)}{\partial w}$
SupTriangle($\triangleleft$)	$\wedge_j(R_{ij} \leftarrow S_{jk})$	$\frac{\partial(\wedge(x\leftarrow w)\vee\vartheta)}{\partial(\wedge(x\leftarrow w))} * \frac{\partial(\wedge(x\leftarrow w))}{\partial(x\leftarrow w)} * \frac{\partial(x\leftarrow w)}{\partial w}$
Square($\square$)	$\wedge_j(R_{ij} \leftrightarrow S_{jk})$	$\frac{\partial(\wedge(x\leftrightarrow w)\vee\vartheta)}{\partial(\wedge(x\leftrightarrow w))} * \frac{\partial(\wedge(x\leftrightarrow w))}{\partial(x\leftrightarrow w)} * \frac{\partial(x\leftrightarrow w)}{\partial w}$

Table 2. Fuzzy operators to be used in RPs definitions

	RP	Definition	Derivative
Implication $a \rightarrow b$	Lukasewicz Kleene-Diens Gaines	$min(1, 1 - a + b)$ $\vee(1 - a, b)$ $\wedge(1, b/a)$	$-\frac{\partial min(1,1-b+a)}{\partial a}$
Equality $a \leftrightarrow b$		$\wedge(a \rightarrow b, b \rightarrow a)$	$\frac{\partial min(a\rightarrow b, b\rightarrow a)}{\partial a} = \frac{\partial min(a\rightarrow b, b\rightarrow a)}{\partial a\rightarrow b} * \frac{\partial a\rightarrow b}{\partial a} + \frac{\partial min(a\rightarrow b, b\rightarrow a)}{\partial b\rightarrow} * \frac{\partial b\rightarrow a}{\partial a}$
t-norm $\wedge(a, b)$	Minimum	$min(a, b)$	$\frac{\partial min(a,b)}{\partial a} = \begin{cases} 1, & a <= b \\ 0, & otros \end{cases}$
	Product	$a * b$	
	Einsten prod.	$(a * b)(2 - a + b - a * b)$	
	Hamacher prod.	$(a * b)(a + b - a * b)$	
	Drastic *t-norm*	$\begin{cases} b \text{ if } a == 1 \\ a \text{ if } b == 1 \\ 0 \text{ otherwise} \end{cases}$	
	Nilpotent	$\begin{cases} min(a, b) \text{ if } (a + b) > 1 \\ 0 \text{ otherwise} \end{cases}$	
s-norm $\vee(a, b)$	Maximum	$max(a, b)$	$\frac{\partial max(a,b)}{\partial a} = \begin{cases} 1, & a >= b \\ 0, & otros \end{cases}$
	Prob. sum	$a + b - a * b$	
	Einsten product	$(a + b)/(1 + a * b)$	
	Hamacher sum	$(a + b - 2 * (a * b))(1 - a * b)$	
	Drastic *s-norm*	$\begin{cases} b \text{ if } a == 0 \\ a \text{ if } b == 0 \\ 1 \text{ otherwise} \end{cases}$	
	Nilpotent	$\begin{cases} max(a, b) \text{ if } (a + b) < 1 \\ 1 \text{ otherwise} \end{cases}$	

4.2 Processing Phase

As described earlier, FRNNs have been extended using different RPs in the processing phase. This work adopt a similar approach using as processing bases the RPs *Circlet*, *Suptriangle*, *Subtriangle* and *Square* (see Table 1). The instances of *t-norm*, *s-norm* and implication considered are shown in Table 2.

5 Experimental Results

The performance of the proposed extension is evaluated in classification problems. Recall that for approaching classification tasks the DMM assigns to an input pattern the label with the highest membership degree. The goal of the experimental evaluation is to show evidence that the alternative RPs considered are competitive with the base FRNN. Accordingly, we evaluate the performance of each possible configuration of an FRNN with the alternative RPs in

a suite of classification problems. The evaluation was performed using 10-fold cross-validation, next we report the achieved accuracy. For the experiments, we considered 9 data sets from the UCI Repository, a summary of the properties of the data sets is given in Table 3. Since one of the strengths of fuzzy algorithms is their tolerance to noise, we also considered data sets with added noise. The noisy databases were taken from the KEEL [1] repository, the noise was introduced with the pattern proposed in [16] with a scheme of Noisy Train - Noisy Test to 20%.

Table 3. Statistics of the data sets. For the '#Features' column R=reals, I=integers, N=nominals.

Data set	# Features (R/I/N)	#samples	#classes
iris	4 (4/0/0)	150	3
car	6 (0/0/6)	1728	4
ecoli	7 (7/0/0)	336	8
pima	8 (8/0/0)	768	2
glass	9 (9/0/0)	214	7
wine	13 (13/0/0)	178	3
heart	13 (1/12/0)	270	2
twonorm	20 (20/0/0)	7400	2
ionosphere	33 (32/1/0)	351	2

Table 4. Performance comparison of the original FRNN (FRNN column) and the proposal of this paper (EFRNN column)

	Noise-free		**Noisy-data sets**	
Dataset	FRNN	EFRNN	FRNN	EFRNN
iris	96.66±3.51	97.33±3.44	82.66±5.62	91.66±7.82
car	78.47±1.57	79.39±2.73	70.02±0.17	73.02±0.32
ecoli	65.79±4.03	74.12±6.39	59.26±6.38	63.06±3.50
pima	74.35±3.16	76.43±1.99	71.22±3.29	72.13±4.56
glass	59.87±6.09	64.06±9.31	55.64±7.06	62.61±5.56
wine	87.09±5.92	97.22±3.92	80.35±8.10	91.01±6.65
heart	76.29±8.76	84.81±8.63	72.96±7.20	79.25±7.02
ionosphere	83.75±5.73	88.04±4.00	81.29±4.02	89.46±4.23
twonorm	90.39±1.88	92.66±1.18	87.10±1.74	89.50±2.44
	79.18±4.24	83.26±4.58	73.38±4.84	78.22±5.76

An FRNN requires the specification of the following parameters: RP for processing, RP for training, number of linguistic terms, type of membership function and number of epochs. The RPs for processing are the result of all possible combinations that arise from replacing the operators (Table 2) in the general definition of the RPs (Table 1). For training the network the listing RPs in Table 1 were taken, combined with the Table 2's operators. The number of linguistic terms can be set to 3, 5 or 7 [12]. The membership functions are used to calculate the membership degree of an input pattern to a class, possible choices are Pi, Triangular and Trapezoidal. These functions are uniformly distributed over the feature interval, expansion factors of $1, 1.25$ and 1.5 can be selected. The number of epochs was set to 10 due to that it was observed that additional training had no significant effect in the performance on the test data sets.

Every possible FRNN configuration, varying the above parameters, was tested on each of the databases. Table 4 shows the results obtained with the base FRNN (column FRNN) and the best FRNN configuration (EFRNN). It can be seen from Table 4 that the best configuration of FRNN found for each data set clearly outperformed the base FRNN. Therefore, we can argue that adding alternative RPs into the training process of FRNNs results in models of better performance; confirming the main hypothesis of this work. It is interesting that the improvements were slightly more important for the noisy data sets. Hence, considering other RPs when training FRNNs reduces the model uncertainty in

[1] `http://sci2s.ugr.es/keel/`

Table 5. FRNN settings that obtained the best results for each data set. The lower section shows the configurations for databases with noise.

Dataset	Membership function	# ling. terms	RP Training | RP Processing	Accuracy
iris	Triangular	3	∘{∧ :Min, ∨ :Max} ◁{∧ :Min, → {KleeneDiens{∧ :HamacherSum}}}	97.33 ± 3.44
car	Trapezoidal	3	◁{∧ :Min, → {Lukasewicz{∧ :BoundedSum}}} ∘{∧ :Min, ∨ :Max}	79.39 ± 2.73
ecoli	Trapezoidal	7	□{↔ {Lukasewicz{∧ :Min}}, ∧ :Min} ∘{∧ :TLukasewicz}, ∨ :EinstenSum}	74.12±6.39
pima	Pi	3	∘{∧ :Min, ∨ :Max} □{↔ {Lukasewicz{∧ :Min}}, ∧ :Min}	76.43±1.99
glass	Trapezoidal	7	▷{∧ :Min, → {Łukasewicz{∧ :Min}}} ∘{∧ :TLukasewicz, ∨ :HamacherSum}	64.06 ± 9.31
wine	Pi	5	□{↔ {Lukasewicz{∧ :Min}}, ∧ :Min} ∘{∧ :HamacherProd, ∨ :EinstenSum}	97.22 ± 3.92
heart	Pi	5	◁{∧ :Min, → {Lukasewicz{∧ :Min}}} ∘{∧ :HamacherProd, ∨ :HamacherSum}	84.81 ± 8.63
ionosphere	Trapezoidal	5	□{↔ {Lukasewicz{∧ :Min}}, ∧ :Min} ∘{∧ :Min, ∨ :EinstenProduct}	88.04±4.00
twonorm	Triangular	7	∘{∧ :Min, ∨ :Max} ∘{∧ :TNilpotent, ∨ :EinstenSum}	92.66±1.18
iris	Pi	3	▷{∧ :Min, → {Lukasewicz{∧ :Min}}} ∘{∧ :Min, ∨ :Prob.Sum}	90.66 ± 7.82
car	Pi	5	□{↔ {Lukasewicz{∧ :Min}}, ∧ :Min} ∘{∧ :TDrastic, ∨ :Max}	73.02±0.32
ecoli	Trapezoidal	5	∘{∧ :Min, ∨ :Max} ∘{∧ :HamacherProduct, ∨ :Max}	63.06±3.50
pima	Trapezoidal	7	∘{∧ :Min, ∨ :Max} ∘{∧ :Min, ∨ :SNilp.}	72.13 ± 4.56
glass	Pi	7	◁{∧ :Min, → {Łukasewicz{∧ :Min}}} ◁{∧ :Min, → {Łukasewicz{∧ :Min}}}	62.61±5.56
wine	Trapezoidal	5	□{↔ {Lukasewicz{∧ :Min}}, ∧ :Min} □{↔ {→ {KD{∨ :Max}}, ∧ :Min}, ∧ :Product}	91.01±6.65
heart	Triangular	7	◁{∧ :Min, → {Lukasewicz{∧ :Min}}} ∘{∧ :Ham.Prod.}, ∨ :ProbSum}	79.25±7.02
ionosphere	Triangular	7	◁{∧ :Min, → {Lukasewicz{∧ :Min}}} ∘{∧ :EinstenProd.}, ∨ :EinsteSum.}	89.46±4.23
twonorm	Trapezoidal	5	□{↔ {Lukasewicz{∧ :Min}}, ∧ :Min} ∘{∧ :HamacherProd, ∨ :EinstenSum}	89.50±2.44

[1] The description of the selected RP is given in *JSON* format, this format consists of an unordered set of name/value pairs. An object begins with '{' (left brace) and ends with '}' (right brace). Each name is followed by ':' and the name/value pairs are separated by ',' (comma).

highly noisy environments. This is a very positive result as the main target of fuzzy algorithms is precisely uncertain environments.

Table 5 shows the best FRNN configuration for each of the considered data sets. It is worth noticing that the Circlet product is highly competitive in most of out experiment on noise free data. In noise free data sets, it was selected in the best FRNN for about half of the data sets. However, for noisy data sets it was in the best model for a two out of the nine data sets only. The diversity of RPs in the training phase is a somewhat expected result (different RPs perform better for different data sets), and confirms our hypothesis and the claims of [3]. This result is evidence supporting the argument that considering alternative RPs for training FRNNs allows us to obtain better performance. Likewise this result reveals that a model selection procedure may be necessary for selecting the adequate RP for training and processing phases.

6 Conclusions and Future Work

We explored the incorporation of alternative RPs into the training phase of FRNNs. A way to add different RPs, which can be more appropriate for a specific application, was outlined. We found that, for some databases, training the

FRNN with the proposed RPs (Circlet, SubTriangle, SupTriangle and Square) outperforms the traditional compositions, proving the validity of our proposal. A limitation of this approach is that, by adding a large number of operators, it makes prohibitive to evaluate all possible combinations, so that as future work we consider to use model selection methods to find the best FRNN configuration for the database under analysis. Also, considering that the outputs of the FRNN represent membership levels of the processed pattern to each class, this approach will be used in problems that give more use to this information.

References

1. Ashtiani, A.A., Menhaj, M.B.: Numerical solution of fuzzy relational equations based on smooth fuzzy norms. Soft Computing 14(6), 545–557 (2010)
2. Bandler, W., Kohout, L.: Mathematical relations, their products and generalized morphisms. Tech. Report Man-Machine Systems Lab. Dept Electrical Engin Univ Essex, pp. 77–3 (1977)
3. Bandler, W., Kohout, L.: Semantics of implication operators and fuzzy relational products. Int'l Journal of Man-Machine Studies 12(1), 89–116 (1980)
4. Barajas, S.E., Reyes, C.A.: Your fuzzy relational neural network parameters optimization with a genetic algorithm. In: The 14th IEEE Int'l Conf. on Fuzzy Systems, pp. 684–689 (2005)
5. Blanco, A., Delgado, M., Requena, I.: Identification of fuzzy relational equations by fuzzy neural networks. Fuzzy Sets and Systems 71(2), 215–226 (1995)
6. Davis, W.L.: Enhancing Pattern Classification with Relational Fuzzy Neural Networks and Square Bk-products. PhD thesis, Tallahassee, FL, USA (2006)
7. de Oliveira, J.V.: Neuron inspired learning rules for fuzzy relational structures. Fuzzy Sets and Systems 57(1), 41–53 (1993)
8. Kohout, L.: Boolean and fuzzy relationsboolean and fuzzy relations. In: Encyclopedia of Optimization, pp. 189–202. Springer (2001)
9. Kohout, L., Kim, E.: The role of bk-products of relations in soft computing. Soft Computing 6(2), 92–115 (2002)
10. Pal, S.K., Mitra, S.: Neuro-fuzzy pattern recognition: methods in soft computing. John Wiley & Sons, Inc. (1999)
11. Pedrycz, W.: Neurocomputations in relational systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 13(3), 289–297 (1991)
12. Reyes, C.A.: On the design of a fuzzy relational neural network for automatic speech recognition. PhD thesis, The Florida State University, Tallahassee, Fl (1994)
13. Reyes, C.A., Bandler, W.: Implementing a fuzzy relational neural network for phonetic automatic speech recognition. In: Fuzzy Modelling, pp. 115–139 (1996)
14. Rosales-Pérez, A., Reyes-García, C.A., Gómez-Gil, P.: Genetic fuzzy relational neural network for infant cry classification. In: Martínez-Trinidad, J.F., Carrasco-Ochoa, J.A., Ben-Youssef Brants, C., Hancock, E.R. (eds.) MCPR 2011. LNCS, vol. 6718, pp. 288–296. Springer, Heidelberg (2011)
15. Zadeh, L.: Similarity relations and fuzzy or. Inform. Sciences 3(2), 177–200 (1971)
16. Zhu, X., Wu, X., Yang, Y.: Error detection and impact-sensitive instance ranking in noisy datasets. In: AAAI, pp. 378–384. AAAI Press (2004)