ADAPTIVE LEARNING ALGORITHM FOR HYBRID FUZZY SYSTEM
HIBRĪDĀS IZPLŪDUŠĀS SISTĒMAS APMĀCĪŠANAI PAREDZĒTS ADAPTĪVS ALGORITMS

Alexander Valishevsky
University of Latvia, Department of Computer Science
Raina bulv. 19, Riga, LV-1586, Latvia
Email: sd80022@lanet.lv, phone: +371-7-583295

Abstract: *In this paper the possibility of improving convergence time of algorithms intended for tuning parameters of fuzzy system with inference mechanism realized with the help of adaptive network is considered. A new algorithm is proposed, which allows to decrease the number of iterations during learning process and to substantially decrease the number of computational operations that have to be performed during single iteration. Furthermore, analytical data is presented and it's shown how to reduce the computational load in the case if the proposed algorithm is being used.*

Keywords: *adaptive learning algorithms, adaptive network, Adaptive Network Based Fuzzy Inference System, neuro-fuzzy systems, Resilient backpropagation.*

1. Introduction

Fuzzy systems are generally used in cases when it's impossible or it's too difficult to define crisp rules that would describe the considered process or system, which is being controlled by a fuzzy control system. Thus, one of the advantages of fuzzy systems is that they allow to describe fuzzy rules, which fit the description of real-world processes to a greater extent. Another advantage of fuzzy systems is their interpretability, it means that it's possible to explain **why** a particular value appeared at the output of a fuzzy system. In turn, some of the main disadvantages of fuzzy systems are that expert input or instructions are needed in order to define fuzzy rules, and that the process of tuning of the parameters of the fuzzy system (e.g. parameters of the membership functions) often requires a relatively long time, especially if there's a high number of fuzzy rules in the system. Both these disadvantages are related to the fact that it's not possible to train fuzzy systems.

A diametrically opposite situation can be observed in the field of neural networks: you can train neural networks, but it's extremely difficult to use a priori knowledge about the considered system and it's almost impossible to explain the behaviour of the neural system in a particular situation.

In order to compensate the disadvantages of one system with the advantages of another system, several researchers tried to combine fuzzy systems with neural networks. A hybrid system named *ANFIS* (*Adaptive-Network-Based Fuzzy Inference System*) has been proposed in [1]. Fuzzy inference in this system is realized with the aid of an adaptive network, which can be considered as a general class of neural networks. Moreover, the author has proposed a training algorithm, which enables to tune the parameters of the fuzzy system. The proposed algorithm is a generalized version of Delta learning rule, which is used to train neural networks and is based on the gradient descent method.

In this paper an alternative adaptive network training algorithm is exposed and, particularly it's described how it can be used to tune parameters of the fuzzy system, which is based on ANFIS architecture. The algorithm is a generalized version of adaptive algorithm Rprop [2]. As can be seen from the results of research exposed in [4], during training of a 'common' neural network, Rprop has a better convergence time than that of Error Backpropagation algorithm and most popular training algorithms with improved convergence time.

Furthermore, the results of this research show that it's possible to reduce the computational load if the proposed algorithm is used, due to the decrease of the number of iterations during the

training process, as well as due to the fact that the algorithm allows to substantially reduce the amount of computational operations that have to be performed during each single iteration.

2. Generalised Adaptive Learning Rule

Let's generalize the Rprop algorithm, which is used to tune weights in a neural network, for the case of adaptive network. This algorithm was introduced in [2]. **Rprop** stands for '**R**esilient back**prop**agation'. This is an adaptive learning scheme, performing supervised batch learning. In this algorithm the partial derivatives are used only to determine the *direction* of the weight step, but not the *size* of the change (the author points to a harmful and unforeseeable influence of the size of the partial derivative on the weight step). Only a sign of the derivative is used to indicate the *direction* of the weight update. The *size* of the weight change $\Delta w_{ij}^{(t)}$ is determined by the so-called 'update-value' $\Delta_{ij}^{(t)}$ (1). The second step of the Rprop learning is to determine the new update-values $\Delta_{ij}^{(t)}$. This is based on a sign-dependent adaptation process (2).

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)}, & if \ \dfrac{\partial E}{\partial w_{ij}}^{(t)} > 0 , \\ +\Delta_{ij}^{(t)}, & if \ \dfrac{\partial E}{\partial w_{ij}}^{(t)} < 0 , \\ 0, & else , \end{cases} \quad (1)$$

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^{+}\Delta_{ij}^{(t-1)}, & if \ \dfrac{\partial E}{\partial w_{ij}}^{(t-1)} * \dfrac{\partial E}{\partial w_{ij}}^{(t)} > 0 , \\ \eta^{-}\Delta_{ij}^{(t-1)}, & if \ \dfrac{\partial E}{\partial w_{ij}}^{(t-1)} * \dfrac{\partial E}{\partial w_{ij}}^{(t)} < 0 , \\ \Delta_{ij}^{(t-1)}, & else , \end{cases} \quad (2)$$

where $\dfrac{\partial E}{\partial w_{ij}}^{(t)}$ denotes the summed gradient information over all patterns of the pattern set (batch learning).

In words, the adaptation rule works as follows: every time the partial derivative of the corresponding weight w_{ij} changes its sign, which indicates that the last update was too big and the algorithm has jumped over a local minimum, the update-value $\Delta_{ij}^{(t)}$ is decreased by the factor η^{-}. If the derivative retains its sign, the update-value is increased by the factor η^{+} in order to accelerate convergence in shallow regions. Additionally, in case of a change in sign, there should be no adaptation in the succeeding learning step.

The author of the algorithm proposes to use the following values of the parameters:

- η^{-} = 0.5, i.e. if a jump over a minimum occurred, we halve the update-value;
- η^{+} = 1.2, as it gave very good results independent of the examined problem.

It's obvious that in order to obtain a generalized version of this algorithm, it's enough to assume that w_{ij} is an arbitrary parameter of a fuzzy system, which has to be tuned. The main difficulty consists in the partial differentiation of $\dfrac{\partial E}{\partial w_{ij}}$. Below the architecture of ANFIS is briefly described and it's shown how to determine partial derivatives, which are needed in order to tune parameters of the given system.

3. ANFIS Architecture

Let's consider a fuzzy system, with two if-then rules of Sugeno type:

$\Re_1$: if x is A_1 and y is B_1 then $z = p_1x + q_1y + r_1$

$\Re_2$: if x is A_2 and y is B_2 then $z = p_2x + q_2y + r_2$

Corresponding ANFIS architecture for this system is shown in Figure 1.

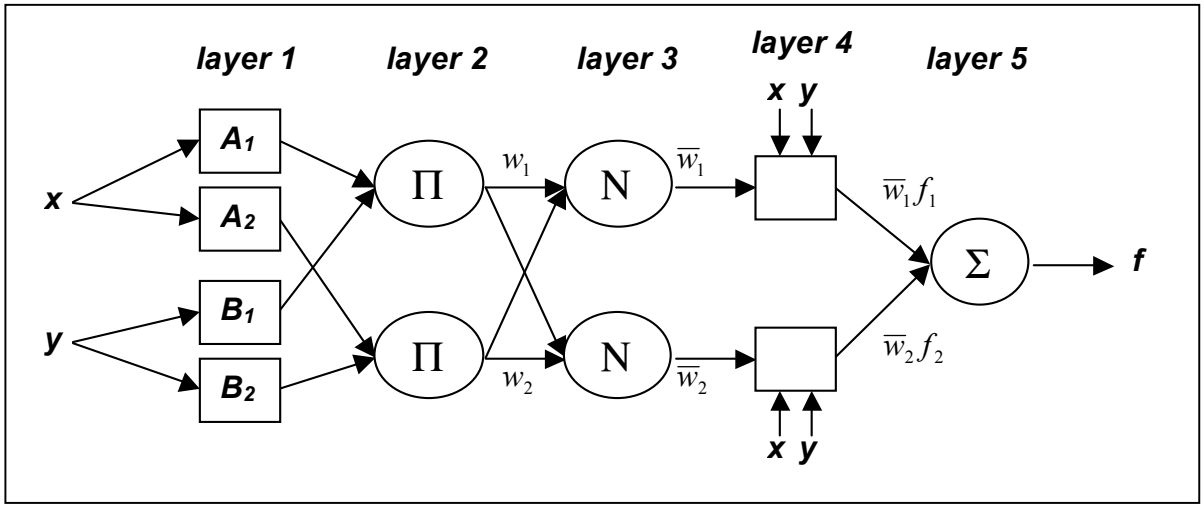

Figure 1. ANFIS architecture for Sugeno type reasoning.

The node functions in the same layer are of the same function family, as described below:

Layer 1: The output of the i-th node of this layer is

$$o_i^1 = \mu_{A_i}(x), \tag{3}$$

where x is the input to the i-th node and A_i is the linguistic label associated with this node function. We'll use bell-shaped membership function, such as:

$$\mu_{A_i}(x) = \exp\left(-\left(\frac{x - c_i}{a_i}\right)^{2b_i}\right), \tag{4}$$

where $\{a_i, b_i, c_i\}$ is the parameter set.

Layer 2: Nodes of this layer multiply the incoming signals and send the product out. Each node output represents the firing strength w_i of the corresponding rule.

Layer 3: The outputs of the nodes in this layer are normalized firing strengths $\overline{w}_i$.

Layer 4: The output of the i-th node of this layer is

$$o_i^4 = \overline{w}_i f_i = \overline{w}_i (p_i x + q_i y + r_i), \tag{5}$$

where $\{p_i, q_i, r_i\}$ is the parameter set.

Layer 5: The single node in this layer computes the overall system output as the summation of all incoming signals:

$$o^5 = \sum_i \overline{w}_i f_i \,. \tag{6}$$

Let's define the measure of error for the k-th learning pattern in the same way, as in common back-prop networks:

$$E_k = \frac{1}{2}(y_k - o_k)^2, \tag{7}$$

where y_k is the desired system output and o_k is the actual system output.

4. Derivation of the Partial Derivatives

Now it's necessary to calculate the partial derivatives of the error function (7) with respect to the parameters of the fuzzy system, which need to be tuned. In other words, for every parameter β_i we have to calculate:

$$\frac{\partial E_k}{\partial \beta_i} = \frac{\partial E_k}{\partial o_k}\frac{\partial o_k}{\partial \beta_i} = -(y_k - o_k)\frac{\partial o_k}{\partial \beta_i}. \tag{8}$$

We'll use the chain rule in order to calculate the partial derivatives. Rumelhart has used this rule during the derivation of equations for the error backpropagation algorithm [3] and it was proposed in [1] for the derivation of the partial derivatives of the error function at the output level w.r.t. adaptive network parameters (e.g. parameters of the membership functions).

We'll consider only the type-3 adaptive network, based on the Sugeno inference mechanism, as the derivation is similar for other types of adaptive networks.

The error rate for consequence parameters can be calculated as follows:

$$\frac{\partial E}{\partial \beta_c} = \frac{\partial E}{\partial o_5}\frac{\partial o_5}{\partial o_4}\frac{\partial o_4}{\partial \beta_c}, \tag{9}$$

where β_c is the consequence parameter and o_i is the output of the *i*-th layer.

The error rate for premise parameters can be calculated as follows:

$$\frac{\partial E}{\partial \beta_p} = \frac{\partial E}{\partial o_5}\frac{\partial o_5}{\partial o_4}\frac{\partial o_4}{\partial o_3}\frac{\partial o_3}{\partial o_2}\frac{\partial o_2}{\partial o_1}\frac{\partial o_1}{\partial \beta_p}, \tag{10}$$

where β_p is the premise parameter and o_i is the output of the *i*-th layer.

$\frac{\partial E}{\partial o_5}$ has been calculated earlier in this paper in (8). The derivation of $\frac{\partial o_j}{\partial o_{j-1}}$, $j = 5,...,2$ follows:

$$\frac{\partial o_5}{\partial o_4} = \frac{\partial\left(\sum f_j \overline{w}_j\right)}{\partial\left(f_i \overline{w}_i\right)} = 1, \tag{11}$$

where $\overline{w}_i$ is the normalized firing strength of the *i*-th rule

$$\frac{\partial o_4}{\partial o_3} = \frac{\partial\left(f_i \overline{w}_i\right)}{\partial\left(\overline{w}_i\right)} = f_i, \tag{12}$$

where *i* is the number of the corresponding rule (or, alternatively, the number of the unit in the 3rd layer)

$$\frac{\partial o_3}{\partial o_2} = \frac{\partial}{\partial w_i}\left(\frac{w_i}{\sum_{j=1}^{n} w_j}\right) = \frac{\sum_{j=1}^{n} w_j - w_i}{\left(\sum_{j=1}^{n} w_j\right)^2}, \tag{13}$$

where i is the number of the corresponding rule (or, alternatively, the number of the unit in the 2nd layer) and n is the total number of rules in the system.

$$\frac{\partial o_2}{\partial o_1} = \frac{\partial}{\partial A_m}\left(\prod_{A_j \in \Re(A_m)} A_j\right) = \prod_{A_j \in \Re(A_m), A_j \neq A_m} A_j \,, \tag{14}$$

where $A_j \in \Re(A_m)$ denotes the fuzzy sets, which make the premise part of the rule containing fuzzy set A_m.

We can aggregate equations (8)-(14) to get a more compact form of the partial derivatives. For the consequence parameter β_c:

$$\frac{\partial E}{\partial \beta_c} = -(d-o)\frac{\partial o_4}{\partial \beta_c}. \tag{15}$$

And for the premise parameter β_p of the membership function of linguistic label A_m:

$$\frac{\partial E}{\partial \beta_p} = -(d-o) f_i \frac{\sum_{j=1}^{n} w_j - w_i}{\left(\sum_{j=1}^{n} w_j\right)^2} \prod_{A_j \in \Re(A_m), A_j \neq A_m} A_j \frac{\partial o_1}{\partial \beta_p}. \tag{16}$$

Now let's calculate the partial derivatives of the output functions w.r.t. their parameters:

$$\frac{\partial o_4}{\partial p_i} = \frac{\partial}{\partial p_i}\left(f_i \overline{w}_i\right) = \frac{\partial}{\partial p_i}\left(\overline{w}_i\left(p_i x + q_i y + r_i\right)\right) = \overline{w}_i x \,, \tag{17}$$

$$\frac{\partial o_4}{\partial q_i} = \frac{\partial}{\partial q_i}\left(f_i \overline{w}_i\right) = \frac{\partial}{\partial q_i}\left(\overline{w}_i\left(p_i x + q_i y + r_i\right)\right) = \overline{w}_i y \,, \tag{18}$$

$$\frac{\partial o_4}{\partial r_i} = \frac{\partial}{\partial r_i}\left(f_i \overline{w}_i\right) = \frac{\partial}{\partial r_i}\left(\overline{w}_i\left(p_i x + q_i y + r_i\right)\right) = \overline{w}_i \,, \tag{19}$$

where i is the number of the corresponding rule.

$$\frac{\partial o_1}{\partial a_{ij}} = \frac{\partial}{\partial a_{ij}}\left(\exp\left(-\left(\frac{x-c_{ij}}{a_{ij}}\right)^{2b_{ij}}\right)\right) = 2\frac{\left(\frac{x-c_{ij}}{a_{ij}}\right)^{2b_{ij}} b_{ij} \exp\left(-\left(\frac{x-c_{ij}}{a_{ij}}\right)^{2b_{ij}}\right)}{a_{ij}}, \tag{20}$$

$$\frac{\partial o_1}{\partial b_{ij}} = \frac{\partial}{\partial b_{ij}}\left(\exp\left(-\left(\frac{x-c_{ij}}{a_{ij}}\right)^{2b_{ij}}\right)\right) = -2\left(\frac{x-c_{ij}}{a_{ij}}\right)^{2b_{ij}} \ln\left(\frac{x-c_{ij}}{a_{ij}}\right) \exp\left(-\left(\frac{x-c_{ij}}{a_{ij}}\right)^{2b_{ij}}\right), \tag{21}$$

$$\frac{\partial o_1}{\partial c_{ij}} = \frac{\partial}{\partial c_{ij}}\left(\exp\left(-\left(\frac{x-c_{ij}}{a_{ij}}\right)^{2b_{ij}}\right)\right) = 2\frac{\left(\frac{x-c_{ij}}{a_{ij}}\right)^{2b_{ij}} b_{ij}\exp\left(-\left(\frac{x-c_{ij}}{a_{ij}}\right)^{2b_{ij}}\right)}{x-c_{ij}}, \qquad (22)$$

where i is the number of the corresponding rule and j is the number of the corresponding linguistic variable in the rule.

As has been stated earlier, the Rprop algorithm uses only the sign of the derivative, so it's possible to substantially simplify the obtained derivatives, as we can reduce them by the elements that don't influence the sign.

Simplified equations are listed below.

For the consequence parameters:

$$\frac{\partial E}{\partial \beta_c} = -(d-o)\frac{\partial o_4}{\partial \beta_c}, \qquad (23)$$

where $\beta_c \in \{p_i, q_i, r_i\}$. The partial derivatives of the output function at the fourth layer w.r.t. the consequence parameters are as follows:

$$\frac{\partial o_4}{\partial p_i} = \overline{w}_i x, \quad (24) \qquad \frac{\partial o_4}{\partial q_i} = \overline{w}_i y, \quad (25) \qquad \frac{\partial o_4}{\partial r_i} = \overline{w}_i, \quad (26)$$

where i is the number of the corresponding rule.

And for the premise parameters:

$$\frac{\partial E}{\partial \beta_p} \approx -(d-o) f_i \frac{\partial o_1}{\partial \beta_p}, \qquad (27)$$

where $\beta_p \in \{a_{ij}, b_{ij}, c_{ij}\}$. The partial derivatives of the output function at the first layer w.r.t. the premise parameters are as follows:

$$\frac{\partial o_1}{\partial a_{ij}} \approx \frac{b_{ij}}{a_{ij}}, \quad (28) \qquad \frac{\partial o_1}{\partial b_{ij}} \approx -\ln\left(\frac{x-c_{ij}}{a_{ij}}\right), \quad (29) \qquad \frac{\partial o_1}{\partial c_{ij}} \approx \frac{b_{ij}}{x-c_{ij}}, \quad (30)$$

where i is the number of the corresponding rule and j is the number of the corresponding linguistic variable in the rule.

5. Several Precautions

In the previous section, equations have been derived, which allow to calculate the partial derivatives of the error function w.r.t. the parameters of the system. It has been shown as well how these equations can be reduced. The equations have been abbreviated by cancellation of the elements which don't influence the sign of the derivative or, more precisely, by the non-negative

elements. It should be noted that these elements could take a zero value, so when implementing the algorithm, checking of the values of these elements should be provided.

The second precaution concerns the b_{ij} parameter. As can be seen from (29), the partial derivative depends on the function of the natural logarithm, which unfortunately is not defined for all possible values of x. The b_{ij} parameter defines the curvature of the membership function, i.e. with its aid it's possible to define the bell-shaped function, which looks more like a trapezoid or a triangular membership function (depending on the value of the parameter). As can be seen, this parameter has only a secondary significance. Thus, to avoid difficulties during the tuning of this parameter, it's advised to tune this parameter intuitively before the network training process begins.

6. Conclusion

The hybrid system 'ANFIS' with inference mechanism based on the adaptive network is being considered in this paper. An important property of this system is that it's possible to tune the parameters of the fuzzy system that has been described with the help of ANFIS. The same methods that are used to tune the parameters are also used to tune the weights in the neural networks. As the ANFIS is a multilayer network, it can be concluded, that the tuning of its parameters is a non-linear task (the parameters of the inner layers can't be expressed linearly). Hence, new algorithms 'inherit' problems that are related to the training algorithms of the multilayer neural networks (e.g. slow convergence, local minima and so on). Moreover, new algorithms require more computational power, as the partial derivatives are more 'bulky' than those in the case of common back-prop networks.

An algorithm, which deals with at least two problems: the convergence speed and the computational load, is exposed in this work. The results of the experiments show that the convergence speed of this algorithm is higher than that of the error backpropagation and most popular algorithms, in which attempts have been made to decrease the convergence time. The new algorithm is adaptive in the sense that the size of the weight-change is tuned during the learning process. Thus, only the sign and not the value of the derivative is used. Hence, it's possible to derive formulae for the determination of the *sign* of the derivatives, which require much less computational power than those for the calculation of the *value* of the derivatives.

Furthermore, an analytical derivation of the partial derivatives of the error function with respect to parameters of the system is exposed in this paper. The chain rule has been used during the derivation. Using this rule and the scheme that has been proposed in this paper, it won't make any difficulties to derive the partial derivations for other types of the fuzzy systems.

Acknowledgement

This research has been financially supported by A/S DATI.

I would like to thank Professor Arkady Borisov from Riga Technical University for his useful comments on the manuscript.

Kopsavilkums

Viens no izplūdušo sistēmu trūkumiem ir tas, ka tās nav iespējams 'apmācīt', un lai noteiktu likumus, pēc kuriem darbojas sistēma, ir nepieciešama eksperta palīdzība. Savukārt, viena no izplūdušo sistēmu priekšrocībām ir tā, ka tām piemīt 'izskaidrošanas spēja', proti, ir iespējams noteikt, *kāpēc* sistēmas izejā parādījās kāda konkrēta vērtība. Pilnīgi atšķirīga situācija ir novērojama neironu tīklu jomā: tos var apmācīt, bet ir gandrīz neiespējami izskaidrot, kāpēc tīkla izejā parādījās kāda vērtība. Daži pētnieki mēģināja savienot neironu tīklu koncepciju ar izplūdušajām sistēmām, lai kompensētu vienas sistēmas trūkumus ar otrās sistēmas priekšrocībām. Rakstā [2] tiek piedāvāta sistēma *ANFIS (Adaptive-Network-Based Fuzzy Inference System),* kurā loģiskais izvedums tiek realizēts ar adaptīva tīkla palīdzību, ko var pieskaitīt pie neironu tīklu vispārīgākas klases.

Darbā tiek apskatīti hibrīdo izplūdušo sistēmu apmācības algoritmi. Tiek aplūkotas sistēmas, kurās izvedums tiek realizēts ar ANFIS arhitektūras palīdzību. Galvenā šīs pieejas priekšrocība ir tāda, ka ir iespējams izveidot algoritmu, ar kura palīdzību būtu iespējams 'pieskaņot' adaptīvā tīkla un, tātad, arī izplūdušās sistēmas parametrus

(piemēram, piederības funkciju parametrus). Tāpat kā neironu tīklu gadījumā, adaptīvā tīkla parametru skaņošanai tiek izmantoti algoritmi, kuri ir balstīti uz gradienta lejupslīdes metodi. No tā, ka adaptīvajam tīklam ir vairāki slāņi, var secināt, ka to apmācīšanas algoritmiem ir raksturīgas tas pašas problēmas, kuras rodas, apmācot daudzslāņu neironu tīklus, proti, zems konverģences ātrums, lokālie minimumi u.t.t. Turklāt jaunie algoritmi prasa lielākus skaitļošanas resursus, jo parciālie atvasinājumi ir sarežģītāki, nekā 'parasto' neironu tīklu gadījumā.

Šajā darbā tiek piedāvāts algoritms, kurš 'cīnās' vismaz ar divām problēmām: ar zemu konverģences ātrumu un ar lielu skaitļošanas slodzi. Piedāvātais algoritms ir algoritma Rprop vispārinājums adaptīvo tīklu gadījumam. Eksperimentu rezultāti liecina par to, ka šim algoritmam ir augstāks konverģences ātrums nekā kļūdas atgriezeniskās izplātīšanas algoritmam un populārākajiem algoritmiem, kuros tika mēģināts paaugstināt konverģences ātrumu [4]. Jaunais algoritms ir adaptīvs tajā nozīmē, ka parametra vērtības izmaiņas lielums tiek pielāgots apmācīšanas laikā. Tādejādi tiek izmantota tikai atvasinājuma zīme, nevis tā lielums. Savukārt, ir iespējams izvest formulas, kuras ir paredzētas atvasinājuma *zīmes* noteikšanai [(23)-(30)] un kuras prasa daudz mazāk skaitļošanas resursus, nekā formulas atvasinājuma *lieluma* izskaitļošanai [(15)-(22)].

Formulas tika vienkāršotas reducējot elementus, kuri neietekmē atvasinājuma zīmi, vai precīzāk – reducējot nenegatīvus elementus. Jāatzīmē, ka šie elementi var pieņemt nulles vērtību, tādejādi, realizējot algoritmu, ir jānodrošina šo elementu vērtību pārbaude.

Darbā ar ķēdes likuma palīdzību tiek noteikti parciāli atvasinājumi, kuri der 2. tipa izplūdušo sistēmu parametru skaņošanai (tās ir sistēmas, kurās tiek izmantoti *Sugeno* tipa likumi). Izmantojot ķēdes likumu un shēmu, kura tiek piedāvāta šajā darbā, var izvest parciālos atvasinājumus cita tipa izplūdušajām sistēmām.

Bibliography

1. Jang J.-S. R. ANFIS: Adaptive-Network-Based Fuzzy Inference System, IEEE Transactions on Systems, Man and Cybernetics, Vol. 23, No. 3, May/June 1993, pp. 665-683.
2. Riedmiller M., Braun H. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm, Proceedings of the IEEE International Conference on Neural Networks, pp. 586-591, San Francisco, 1993.
3. Rumelhart D.E., McClelland J.L., eds. Parallel Distributed Processing, Vol. 1, pp. 318-362, MIT Press, Massachusetts, 1986.
4. Valishevsky A. Comparative Analysis of Different Approaches towards Multilayer Perceptron Training, Scientific Proceedings of Riga Technical University, 2001.