COMPUTER-AIDED DESIGN SYSTEMS

UDC 004.8 (045)

O. I. Chumachenko

DEEP LEARNING CLASSIFIER BASED ON NEFPROX NEURAL NETWORK

Technical Cybernetic Department, National Technical University of Ukraine
"Ihor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
E-mail: chumachenko@tk.kpi.ua

Abstract—*It is proposed a new class of fuzzy classifiers. It is a deep learning classifier based on NEFPROX neural network. The pre-learning is supplied with help of Restricted Boltzman Machine.*

Index Terms—Fuzzy classifiers; deep learning; NEFPROX neural network Restricted Boltzman Machine.

I. INTRODUCTION

The neural networks are currently used for the solution of various kinds of problems, such as approximation, classification, pattern recognition, prediction. Along with the classical approach, there are a number of methods for these problems solution, to simplify and speed up the computations. One of such method is the transition to the fuzzy logic. Fuzzy logic systems transform the numerical inputs of neural networks into data of fuzzy nature using the fuzzification process. The main element of fuzzification is a membership function μ. Its value determines whether the numerical value of the input "low", "normal" or "great". The most common types of membership functions are triangular, trapezoidal, bell-shaped and Gaussian function. Further there is the aggregation of obtained results using the fuzzy rules, logical conclusion and defuzzification – transform of fuzzy values into numeric format [5].

NEFPROX network belongs to a class of three-layer fuzzy perceptrons, NEFPROX neural network has a three-layer serial structure shown in Fig. 1.

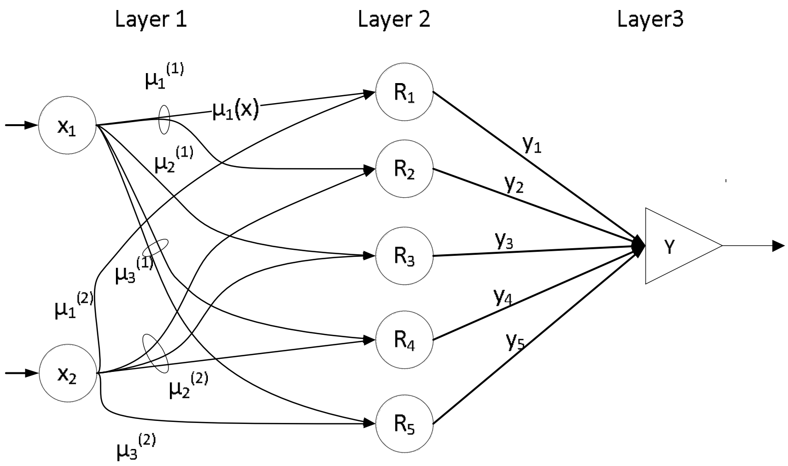

Fig. 1. Network structure NEFPROX

1) The first layer is the input data and does not change their values.

2) The neurons of the hidden layer (Layer 2) contain fuzzy rules, for example, fuzzy rule R_j takes the form: if x_1 belongs $\mu_k^{(1)}$, x_2 belongs $\mu_l^{(2)}$, … , x_n belongs $\mu_p^{(n)}$, the output R_j will be equal to 1, where $\mu^{(1)} \dots \mu^{(n)}$ are membership functions.

In general, the membership function for the network NEFPROX bell-shaped function of the following form

$$\mu(x) = \begin{cases} \dfrac{x-a}{b-a}, & a < x < b, \\ \dfrac{c-x}{c-b}, & b < x < c, \\ 0, & \end{cases} \qquad (1)$$

where c, a, b are adjustable parameters during training process. The number of membership functions is given in an arbitrary manner, increasing the number of membership functions leads to the improved accuracy.

The formula for calculating the output usually looks like this:

$$f(x) = \lambda_1 x_1 + \lambda_2 x_2 + ... + \lambda_n x_n, \quad (2)$$

3) The third layer is composed of an adder, which calculates output NEFPROX neural network. The output value is calculated according by the equation

$$Y = \sum R_j, \quad (3)$$

where R_j is the output of the second layer.

II. Problem Statement

The standard approach to learning is based on the method of back propagation [1], however, a large number of adjustable parameters lead to a halt in the algorithm stop in local extremums, that influences badly which adversely affects the accuracy of the network [1].

To improve the accuracy of the network uses the concept pre-learning [2] used in training deep neural networks, for which the basic learning algorithm is a method of backpropagation. Pre-learning is done by constructing autoassociator – neural network, the output of which is to be closest to the value of the input data. Autoassociator trained teachers without the inputs identified in the training set, then the weighting coefficients autoassociator transferred to the main network and continue learning by back propagation. Thus, there is an initial setup that allows you to be as close as possible to the global extremes, which increases the accuracy of the network. The most common autoassociator are Restricted Boltzman Machine (RBM) and autoencoders [3]. In this paper we used the RBM is limited.

II. Problem Solution

Since pre-learning assumes autoassociator structure (the number of neurons and interneuronal connections, as well as activation of neuronal function) coincides with the structure of the network that pre-learning [4]. For NEFPROX pre-learning network uses an approach similar [9]. It includes calculating the membership functions in a single layer and neurons of the initializing layer coefficients using Restricted Boltzman Machine (RBM). Process pre-learning NEFPROX network is shown in Fig. 2.

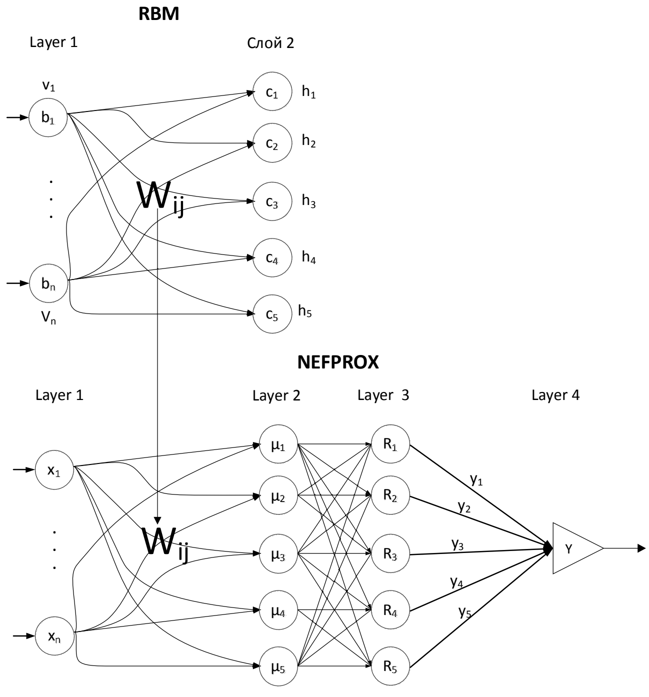

Fig. 2. Pre-learning NEFPROX neural network

IV. Network Training Algorithm

Step 1. Structural adjustment network

1) For all possible combinations of values $\mu_j(x_i)$ it is created the rules R_k. During training for sigmoidal functions of the form (1) it is necessary to adjust two parameters: slope w and the bias c.

2) For all $\mu_j(x_i)$ it is set the initial values. The initial values of membership functions must correspond to the rule

$$\forall x_k \in O(x_i) : \exists \mu_j(x_i) \neq \pm 1,$$

where $O(x_i)$ is the domain of input variable definition. That is, for each value of the input variable domain of definition, there is a corresponding value of the membership functions other than ± 1. An example of the initial distribution of membership functions for the domain [-1,1] is shown in Fig. 3.

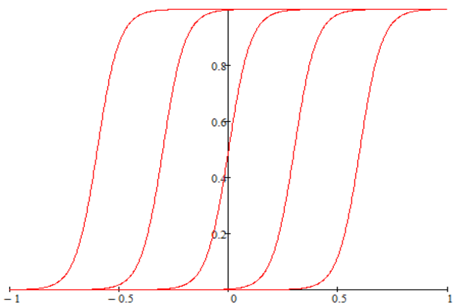

Fig. 3. The initial distribution of the membership functions

Step 2. Parametric adjustment of the RBM is made by the algorithm independent differences

1) The initial values v_0 set equal to the input data.

2) Calculate the probability that the state of the neuron of the hidden layer is equal to 1:

$$p(h_j = 1 | v) = \frac{1}{1+\exp\left(\sum_{i=1}^{m} w_{ij} v_i + b_i\right)}, \quad (4)$$

where c_i is the bias of the hidden layer.

3) Assign the value of the hidden layer neuron based on the probability (if the probability of 0.9, then with probability 90% of neuron will be 1, and with a probability of 10% – 0).

4) Calculate the probability that the state of the neuron of the visible layers will be equal to 1:

$$p(v_i = 1 | h) = \sigma\left(\sum_{j=1}^{m} w_{ij} h_j + c_j\right), \quad (5)$$

where c_j is the bias of the visible layer; σ is the function

$$\sigma = \frac{1}{1+e^{-x}}. \quad (7)$$

5) Assign neuron layer visibility based on the probability value.

6) Repeat *step 2* if the iteration number is less than *k*.

7) Calculate the probability of the hidden layer.

A. Initial parameter values

1) The states of the hidden layer neurons are given by input data.

2) The initial value of neural connections weights are given as a small value of about 0 to Gause distribution.

3) The biases of the visible layer are given.

B. Parameters modification

1) Synaptic connections

$$w_{ij}(t+1) = w_{ij}(t) + \alpha\left[v_i(0)h_j(0) - v_i(k)h_j(k)\right].$$

2) Biases of the visible layer

$$b(t+1) = b_i(t) + \alpha\left[v_i(0) - v_i(k)\right].$$

3) Biases of the hidden layer

$$c(t+1) = c_j(t) + \alpha\left[h_j(0) - h_j(k)\right],$$

where α is the training speed.

Step 3. Parametric fuzzy network

The parameters obtained by learning MBP transferred to the network as follows:

1) The weighting coefficients $w_{ij}^{\mathrm{NEFPROX}} = w_{ij}^{\mathrm{RBM}}$

2) The bias of the second layer neurons $\beta_j^{\mathrm{NEFPROX}} = b_j^{\mathrm{RBM}}$.

For training example, we find a difference between the reference value and the value obtained as a result of the network functioning

$$\Delta_c = y_e - y_r.$$

3) We calculate Δ_R for the rule.

$$\Delta_R = U_R(1-U_R)\sum_{c\in U_3} W(R,c)\Delta_c$$

where U_R is the output of the rules layer; $W(R,c)$ is the connection weight of rule neuron with class neuron.

4) We find x' which is the minimum of function $W(x',R)U_{x'}$

$$W(x',R)U_{x'} = \min\{W(x,R)U_x\}$$

where $W(x',R)U_{x'}$ is the membership functions.

5) For the membership function it is determined the values on which it is necessary to change the parameters values

$$\Delta_w = -\sigma\Delta_R \frac{-xe^{wx'+\beta}}{(e^{wx}+e^{\beta})}, \quad \Delta_\beta = -\sigma\Delta_R \frac{e^{wx'+\beta}}{(e^{wx}+e^{\beta})^2},$$

where σ is the training speed.

6) Change the parameters, according to step 5 and calculate the rule error

$$E = U_R(1-U_R)\sum_{c\in U_3}(2W(R,c)-1)\Delta_c.$$

The complex criterion of stop is:

1) $\sum_{i=1}^{n} \Delta_{c_i} \le \Delta_c^*$, i the number of examined examples; Δ_c^* is the threshold value; n is the size of examined sample.

2) The Δ_{c_i} doesn't change during some iterations

$$\left|\Delta_{c_i} - \Delta_{c_j}\right| \le \varepsilon,$$

where i, j is the number of iterations $i \ne j$.

V. CONCLUSIONS

The opportunity to train NEFPROX neural network according to the paradigm pre-learning autoassociator, these results demonstrate improved accuracy on the test sample.

REFERENCES

[1] D. Nauck, U. Nauck, and R. Kruse, "Generating classification rules with the neurofuzzy system NEFCLASS." *Biennial Conference of the North American, Fuzzy Information Processing Society*, 1996, pp. 466–470.

[2] Y. Bengio, "Learning deep architectures for AI." *Foundations and trends® in Machine Learning*, 2(1), pp. 1–127, 2009.

[3] Y. Bengio, *Deep learning of representations: Looking forward, Statistical Language and Speech Processing*. Springer Berlin Heidelberg, 2013, pp. 1–37.

[4] J. Schmidhuber, "Deep learning in neural networks: An overview." *Neural Networks*, 61, pp. 85–117. 2015.

[5] L. A. Zadeh, "Fuzzy algorithms." *Information and Control*. 12 (2), pp. 94–102, 1968.

[6] G. Hinton, *A Practical Guide to Training Restricted Boltzmann Machines*, 2010, pp. 3–17.

[7] A. Fischer, and C. Igel, *An Introduction to Restricted Boltzmann Machines*, 2011, pp. 1–23.

[8] M. Nielsen, (viewed on September 20, 2016). Using neural nets to recognize handwritten digits [Electronic resourse] – Electronic data.– Mode of access: http://neuralnetworksanddeeplearning.com/chap1.html

[9] O. I. Chumachenko. Deep Learning Classifier Based on NEFCLASS Neural Network // Electronics and Control Systems, N 3(49) – Kyiv: NAU, 2016. – pp. 79–83.

Received September 18, 2016

Chumachenko Olena. Candidate of Science (Engineering). Associate Professor.
Technical Cybernetic Department, National Technical University of Ukraine "Ihor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.
Education: Georgian Politechnic Institute, Tbilisi, Georgia, (1980).
Research area: system analysis, artificial neuron nets.
Publications: more than 50 papers.
E-mail: chumachenko@tk.kpi.ua

О. І. Чумаченко. Класифікатор глибокого навчання на основі нейронної мережі NEFPROX
Запропоновано новий клас нечітких класифікаторів. Це класифікатори глибокого навчання на основі нейронної мережі NEFPROX. Попереднє навчання забезпечується за допомогою обмеженої машини Больцмана.
Ключові слова: нечіткі класифікатори; глибоке навчання; нейронна мережа NEFPROX; обмежена машина Больцмана.

Чумаченко Олена Іллівна. Кандидат технічних наук. Доцент.
Кафедра технічної кібернетики, Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського», Київ, Україна.
Освіта: Грузинський політехнічний інститут, Тбілісі, Грузія, (1980).
Напрям наукової діяльності: системний аналіз, штучні нейронні мережі.
Кількість публікацій: більше 50 наукових робіт.
E-mail: chumachenko@tk.kpi.ua

Е. И. Чумаченко. Классификатор глубокого обучения на основе нейронной сети NEFPROX
Предложен новый класс нечетких классификаторов. Это классификаторы глубокого обучения на основе нейронной сети NEFPROX. Предварительное обучение обеспечивается с помощью ограниченной машины Больцмана.
Ключевые слова: нечеткие классификаторы; глубокое обучение; нейронная сеть NEFPROX; ограниченная машина Больцмана.

Чумаченко Елена Ильинична. Кандидат технических наук. Доцент.
Кафедра технической кибернетики, Национальный технический университет Украины «Киевский политехнический институт им. Игоря Сикорского», Киев, Украина.
Образование: Грузинский политехнический институт, Тбилиси, Грузия, (1980).
Направление научной деятельности: системный анализ, искусственные нейронные сети.
Количество публикаций: более 50 научных работ.
E-mail: chumachenko@tk.kpi.ua