Deep learning neural networks in fuzzy modeling

D Sotvoldiev, D T Muhamediyeva, Z Juraev

Tashkent University of Information Technologies named after Muhammad al-Khwarizmi, Amir Temur street, 108, 100200, Tashkent, Uzbekistan

Abstract. Deep learning neural networks have achieved remarkable success in various fields of application of artificial intelligence. This technology, which grew out of studies of artificial neural networks, has shown superior performance compared to other machine learning algorithms in areas such as image and voice recognition, natural language processing, and others. The use of deep learning neural networks in the construction of a fuzzy model is proposed.

1.Introduction

Digital data in all shapes and sizes is growing exponentially. According to the US National Security Agency, in 2011, digital in just five years, the amount of information has grown nine-fold [1.2] and by 2020 its amount in the world is expected to reach 35 trillion gigabytes [3]. High demand for the study and analysis of big data (BigData) stimulated the emergence and use of deep learning algorithms (DeepLearning, DL). DL has achieved tremendous success in a wide range of applications, computer games, speech recognition, computer vision, natural language processing, etc. [4]. According to Gartner, in the top ten technology trends in 2018, artificial intelligence technologies are on the top line [5]. Over the past decade, deep learning neural networks have achieved remarkable success in various fields of application of artificial intelligence. This technology, which grew out of studies of artificial neural networks, has shown superior performance compared to other machine learning algorithms in areas such as image and voice recognition, natural language processing, and others.

DL is a class of machine learning algorithms that uses artificial neural networks (ANNs) with many layers of nonlinear block processing to train data. The earliest TIN was developed in 1943 [6], when Warren McCallough and Walter Pitts proposed a computational model for NS based on mathematics and algorithms called threshold logic.

Input variables are called input nodes, and variables are converted through hidden nodes, and at the end, the output values are calculated on the output nodes.

2. Neural networks for deep learning

Mathematical model of a neuron is described by the following expression:

$$I = \sum w_i x_i .$$

Output signal of the neuron is determined by the nonlinear function F

$$Y = F(I - \theta),$$

where θ - hreshold of the neuron. Usually, the simplest nonlinear functions of the following types are used as the F function:

a) binary (threshold)

$$y = \begin{cases} 1, & at\ I > \theta \\ 0, & at\ I \le \theta. \end{cases}$$

b) sigmoid

$$y = \frac{1}{1 - e^{-(I-\theta)}} .$$

We only note that the main difference between DL and traditional neural networks is the scale and complexity. DL uses more hidden layers, while ANNs usually have only one or two hidden layers. Deep learning networks use many nodes and hidden layers due to the advent of more powerful technology (GPUs). Submit some popular architectures used in DL, a fully connected deep neural network (DNN), which contains several hidden layers, and each layer contains hundreds of nonlinear neurons [7,8].

3. Popular architectures used in DL. Fully connected and hierarchical deep neural network (DNN)

Neural network is a collection of individual neurons connected in a specific structure. The computing power of the network, the tasks that it can solve, are set precisely by these connections. Connections connect the inputs of some neurons with the outputs of others, and their "strength" is given by weighting factors (or, simply, weights). Thus, the strength of the influence of the behavior of one neuron on the behavior of another is determined by the corresponding weight of the connection. Therefore, neural systems are often also called connectionist (from the word connection, connection).

The order of connections in the network is its architecture [1,2]. There are two types of neural network architecture: full connected and hierarchical networks. As is known from graph theory, in a fully connected architecture, all network elements are connected to each other. In terms of neural networks, this means that the output of each neuron is connected to the inputs of all other neurons and its inputs are connected to the outputs of the remaining elements. In addition, the output of each neuron is connected to its input (the so-called "self-to-self communication"). The number of connections in a fully connected network containing N neurons is equal, since N connections come from each node.

In a hierarchical architecture, you can select groups of neurons located on separate layers or levels. Each layer of neuron is associated with each of the neurons of the previous and next layers. The neurons of the input layer receive signals from the external environment and distribute them among the neurons of the subsequent layer. The outputs of the neurons of the output layer enter the external environment. Layers located between the input and output are called intermediate or hidden (since they have no direct connection with the external environment).

In the direction of signal transmission in the network, one can distinguish networks without feedback, or non-recurrent (feed-forward) networks and networks with feedback, or recurrent (feed-back) networks.

In networks without feedbacks, neurons of one layer receive signals only from the external environment or from neurons of the previous layer and transmit signals either to the external environment or to the inputs of the neurons of the next layer.

In recurrent networks, neurons of a particular layer can also receive signals from themselves and other neurons located on the same layer. Thus, in contrast to non-recurrent networks, the values of the output signals of the neurons of the recurrent network are determined not only by the values of the current signals at the inputs of the neurons and weights of the corresponding connections, but also by the outputs of some neurons at the previous time point. This means that such a network has memory elements, which allows you to store information about the states of the outputs for a certain time.

In the case when the recurrent network has inhibitory connections (ie, connections with negative weights) with the neurons of its layer, it is called a network with lateral inhibition.

Recurrent networks behave quite differently. When the vector is fed to the network input, the states of the neurons are determined, but then, because the outputs of the neurons have feedbacks, a new vector again arrives at their inputs, and the states change again. The concept of stability is connected with recurrent networks. A network is considered stable if, after a finite number of iterations, neurons

assume states that do not change later. When a vector is fed to the input of stable recurrent networks, the output signals of neurons are generated, which are then fed back to the inputs, again generating a new state vector, but as the number of iterations grows, the number of changes in node states decreases until the network is in the final state. Networks without feedbacks are always stable, since when a single vector is fed to an input, network nodes can change their state only once, due to the constancy of the inputs of neurons.

Unstable networks do not reach the final state in a finite number of iterations. In them, when the input vector is fed, output signals are generated, which are fed again to the inputs and thus cause new state changes, and this process continues indefinitely, without being established in some unchanged state.

Relaxation in unstable neural networks, also called chaotic relaxation, is of great interest to researchers, but the problems of chaotic relaxation are not covered in this paper.

For a long time, the question of how to judge the stability of the network should remain open. This hindered the development of work in this direction. However, thanks to the proof of a theorem defining a subset of recurrent networks that are stable, this issue has been resolved, and today many scientific developers are engaged in the study of complex recurrent networks.

One of the largest researchers of neural networks is Hopfield, who proposed a special kind of networks, which are called so - Hopfield networks.

Thus, a neural network that has n outputs that accept binary values can be in states. Each of these states is one of the vertices of the hypercube in n-dimensional space if the values of neuron activity levels are plotted along the coordinate axes. When the input vector is fed, the network goes from state to state, i.e. moves along the vertices of the hypercube until it reaches the final state. Stable conditions correspond to certain values of the input vectors, weight coefficients and thresholds. Since in the same network the weights and threshold values remain constant (the learning mode is not considered here), the stable states are completely determined by the current input vectors.

Kohonen and Grossberg [6] showed that if the weights matrix of a recurrent network is symmetric with respect to the main diagonal containing zeros, then this network is stable. This statement is proved as follows. Suppose that there is a function that reduces its value with each change in the network state. Obviously, this function should decrease until it reaches zero. As the function itself, you can take the following Lyapunov function

$$E = -\frac{1}{2}\left(\sum_i \sum_j w_{ij} y_i y_j - \sum_j x_j y_j + \sum_j \theta_j y_j \right). \quad (1)$$

Here E is the so-called "energy" of the network: x_j - j-th component of the input vector supplied from the outside; θ_j - neuron threshold j;

Considering the expression (1), we write the formula for the change in energy caused by a change in the state of the neuron j

$$\Delta E = -\left[\sum_{i \neq j} (W_{ij} Y_i) + X_j - \theta_j \right] \Delta y_j . \quad (2)$$

Consider the expression (2). If the total weighted input of the neuron exceeds the threshold value, then it switches to the active state, which means that with a positive value, the expression in square brackets y_j takes the value "1".

4. Stochastic deep neural networks

In the real world one often comes across stochastic processes. The difficulty, and sometimes the impossibility of taking into account all the factors that influence the behavior of a real system, forces

us to consider such systems as stochastic. The most famous class of stochastic neural networks are the so-called Boltzmann machines [6]. They are built on the principle of Hopfield networks, i.e. they use a symmetric weights matrix, with the only difference being that the rule for activating neurons here is stochastic, and not deterministic. This means that instead of the function that directly determines the level of the output signal of a neuron with given input signals, the relation that determines the probability of a neuron to trigger is used here.

Concept of energy and stability was connected with the networks of Hopfield. Consider the following relationship, which determines the energy of the network

$$E = \sum_{i<j} w_{ij} y_i y_j + \sum_i \theta_i y_i .$$

y_i and y_j - activity levels of the i-th and j-th neurons, respectively; θ_i - threshold of i-th neuron. The activation of the neuron k leads to a decrease in the total energy of the network by

$$\Delta E_k = \sum_i w_{ki} - \theta_k .$$

In stochastic networks, the activation rule is given by some probability density function f

$$p_k = f(\Delta E_k) = f\left(\sum_i w_{ki} y_i - \theta_k \right),$$

where p_k - probability of k-th neuron triggering.

As the function f, the threshold function cannot be used, since in this case, the network becomes deterministic. It is also unacceptable to use the linear function and the hyperbolic tangent, since the areas of their values fall outside the limits of the range [0,1], in which the probability value can be. Therefore, the sigmoid function is often used as a function of the probability distribution density.

$$p_k = \frac{1}{1 + e^{-\Delta E_k / T}},$$

where T is some parameter that determines the level of randomness in the network. The parameter T is called “temperature”; this is because a parallel can be drawn between the Boltzmann machine and the thermodynamic system.

5. General algorithm for deep learning neural networks in fuzzy modeling

General fuzzification rule for learning deep neural networks is considered. The overall output of the considered deep learning neural networks in fuzzy modeling is defined as

$$\tilde{Y}_i = f\left[\sum_{j=1}^{n} \tilde{k}_j \tilde{z}_{ij} \right],$$

$\tilde{z}_{ij}$ output of the jth neuron of the hidden layer when the vector arrives at the network input $\tilde{X}_i = (\tilde{x}_{i1}, \tilde{x}_{i2}, ..., \tilde{x}_{in})$, $\tilde{k}_j$ - a fuzzy weight coefficient of connection between the j-th neuron of the middle layer and the output (m + 1) -th, f is the activation function, which in this work is taken as

$$f(y) = [1 + e^{-y}]^{-1}.$$

$\tilde{z}_{ij}$ defined as

$$\tilde{z}_{ij} = f\left[\sum_{l=1}^{L} \tilde{x}_{il}\tilde{w}_{lj}\right],$$

where $\tilde{w}_{ij}$ - fuzzy connection weighting factor between the i-th neuron of the input layer and the j-th neuron of the hidden layer.

Fuzzy error between the current output of deep learning neural networks and the desired output is defined as

$$\tilde{E} = \frac{1}{2}\sum_{l=1}^{L} (\tilde{Y}_l - \tilde{Y}_l^*)^2. \quad (3)$$

$\tilde{Y}_i^*$ - desired fuzzy output value HHC_3. Question of determining the learning algorithm HHC_3, consisting in determining such values $\tilde{w}_{ij}$ and $\tilde{k}_j$, which provide minimization (3), i.e. approximation $\tilde{E}$ to zero.

However, since in the calculation $\tilde{E}$ fuzzy subtraction is used, then even $\tilde{Y}_l = \tilde{Y}_l^*$ for all l $\tilde{E}$ will not be zero. Consequently, it is necessary to form the rules for stopping computations in neural networks of deep learning. Imagine that the carrier of a fuzzy set $\tilde{Y}_l$ is the interval $\left[\tilde{Y}_{l1}, \tilde{Y}_{l2}\right]$, $1 \le l \le L$. Then if $\tilde{Y}_l = \tilde{Y}_l^*$, then for $\tilde{E}$ the carrier is the interval $[-\lambda, \lambda]$, where

$$\lambda = \frac{1}{2}\sum_{l=1}^{L} (Y_{l1}^* - Y_{l2}^*)^2.$$

If $\varepsilon > 0$ denotes an acceptable deviation from $\tilde{E}$, when $\tilde{Y}_l = \tilde{Y}_l^*$, This stopping rule can be formed as follows [31,48]. Search Iterations $\tilde{w}_{ij}$ and $\tilde{k}_j$ need to stop when $\tilde{E}$ is inside the set

$$\Omega = [-\lambda - \varepsilon, \lambda + \varepsilon] \times [0,1].$$

Let us proceed to the formation of the learning algorithm itself. Learning Rules for $\tilde{k}_j$ will be like

$$\tilde{k}_j(q+1) = \tilde{k}_j(q) + \eta\Delta\tilde{k}_j(q), \quad 1 \le j \le m; \quad q = 1,2,...,$$

$$\Delta\tilde{k}_k = \frac{\partial\tilde{E}}{\partial\tilde{k}_j},$$

$$\frac{\partial\tilde{E}}{\partial\tilde{k}_j} = \sum_{l=1}^{L} \left(\tilde{Y}_l - \tilde{Y}_l^*\right)\left(\tilde{Y}_l\right)\left(1 - \tilde{Y}_l\right)\tilde{z}_{lj},$$

η - learning speed.

To determine the weights $\tilde{w}_{ij}$ you can use the rules

$$\tilde{w}_{ij}(q+1) = \tilde{w}_{ij}(q) + \eta\Delta\tilde{w}_{ij}(q),$$

$$\frac{\partial \tilde{E}}{\partial \tilde{w}_{ij}} = \sum_{l=1}^{L} (\tilde{Y}_l - \tilde{Y}_l^*)(\tilde{Y}_l)(1-\tilde{Y}_l)\tilde{k}_j \frac{\partial \tilde{z}_{ij}}{\partial \tilde{w}_{ij}},$$

$$\frac{\partial \tilde{z}_{lj}}{\partial \tilde{w}_{ij}} = (\tilde{z}_{lj})(1-\tilde{z}_{lj})\tilde{x}_{li}.$$

6. Computational experiment

Above algorithm is based on fuzzy arithmetic and fuzzy analysis [9-11].

Solved the problem of classification and evaluation using the developed program, conducted a comparative analysis between the results of the proposed and known algorithms.

Table 1 compares the results of solving some model problems on the basis of various known and proposed algorithms.

Table 1

Results of the proposed and existing algorithms

Task	Neuro-fuzzy deep learning networks	SVM	1NN	KNN	Conventional RBF network
Glass	87.85	71.50	72.01	72.01	69.16
Iris	98.3	97.33	96.00	95.33	95.33
Wine	98.88	99.44	95.52	96.07	98.89

7. Conclusion

Thus, deep learning neural networks are universal approximants and classifiers, i.e. they can approximate any function with any accuracy when setting a sufficient number of neurons. And there is no need to choose the type of model.

In neural approximation, pairs of observed values of variables and values of functions for network training are considered. After training, the network can perform the mappings with a high degree of accuracy. The combination of variables sets the points in the N-dimensional space, where N is the dimension of the vector of variables of the function. Each of these points is associated with a point in the one-dimensional function value space. The more pairs of type (X, Y) are selected, the more accurate the approximation.

In practice, this allows the identification of objects for which it is difficult to build an exact mathematical functional description.

References

[1] Xu Y. et al. 2011 *J. Chem. Inf. Model.* **57** p 2672
[2] Lenselink E B 2017 J. *Cheminformatics* **9** p 45
[3] Gaulton A 2012 J.*Nucleic Acids Res.* **40** p 1100
[4] Howard J 2013 *In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* p 1135
[5] Papadatos G 2015 *J. Comput,* T **29** p 885
[6] Srivastava N 2014 *Machine Learning Res* T **15** p 1929
[7] Wan L 2013 *In Proceedings of the 30th International Conference on Machine Learning,* T **28** p 1058
[8] Nair V and Hinton G E 2010 *In Proceedings of the 27th International Conference on International Conference on Machine Learning* p 807
[9] Muhamediyeva D K 2019 *IOP Conf. Journal of Physics*: *Conf. Series 1210*
[10] Muhamediyeva D T 2019 *IOP Conf. Series: Journal of Physics: Conf. Series 1210*

[11] Muhamediyeva D T 2019 *International Journal of Mechanical and Production Engineering Research and Development* **9** *Issue 2* p 649