References:

1. Vakalyuk T. A., Bolotina V. V., Bailyuk E. M., Pokotilo O. A. Review of game online services for learning programming languages. *Innovative pedagogy*. 2020. Vol. 1. No. 22. P. 192–198.

2. Online compiler what it is. Top online compilers. GitJournal. URL: https://gitjournal.tech/podborka-onlajn-kompiljatorov-chto-jeto-kak-oni-rabotajut-i-kakoj-vybrat

3. Programming practice with Repl.it – IDE-based browser and compiler / Coding. The best lessons in web development. URL: https://ua.phhsnews.com/articles/coding/practice-programming-with-repl-it-a-browser-based-ide-and-compiler.html

DOI https://doi.org/10.30525/978-9934-26-277-7-109

NEURAL NETWORKS: BASIC PROVISIONS

НЕЙРОННІ МЕРЕЖІ: ОСНОВНІ ПОЛОЖЕННЯ

Kodirov E. S.
Assistant Teacher at the Department of Internet and Information Communication
Korea International University in Fergana
Fergana, Uzbekistan

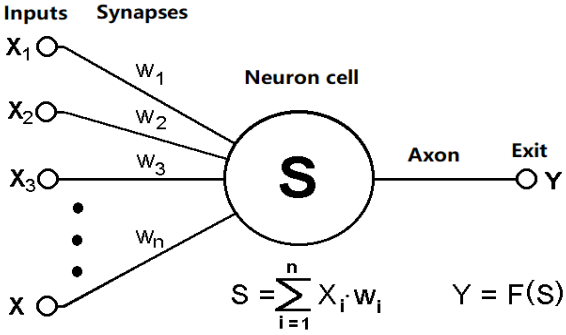
Кодиров Є. С.
асистент викладача кафедри Інтернету та інформаційних комунікацій
Корейський міжнародний університет в Фергані
Ферган, Узбекистан

In recent decades, a new applied area of mathematics has been rapidly developing in the world, specializing in artificial neural networks (NNs). The relevance of research in this direction is confirmed by the mass of various applications of NN. These are automation of image recognition processes, adaptive control, approximation of functional, forecasting, creation of expert systems, organization of associative memory and many other applications. With the help of neural networks, one can, for example, predict the performance of the stock market, perform recognition of optical or audio signals, create self-learning systems that can control a car when parking, or synthesize speech from text [1].

A wide range of tasks solved by the NN does not currently allow the creation of universal, powerful networks, forcing the development of

specialized NNs that operate according to various algorithms. NN models can be software and hardware versions. In what follows, we will focus mainly on the first type.

Despite significant differences, individual types of NN have several common features. Firstly, each NN is based on relatively simple elements (cells) of the same type in most cases that imitate the work of brain neurons. Further, a neuron will mean an artificial neuron, that is, a NN cell. Each neuron is characterized by its current state, by analogy with the nerve cells of the brain, which can be excited or inhibited. It has a group of synapses – unidirectional input connections connected to the outputs of other neurons, and also has an axon – an output connection of a given neuron, from which a signal (of excitation or inhibition) enters the synapses of the following neurons. The general view of a neuron is shown in Pic 1. Each synapse is characterized by the value of the synaptic connection or its weight w, which is physically equivalent to electrical conductivity.



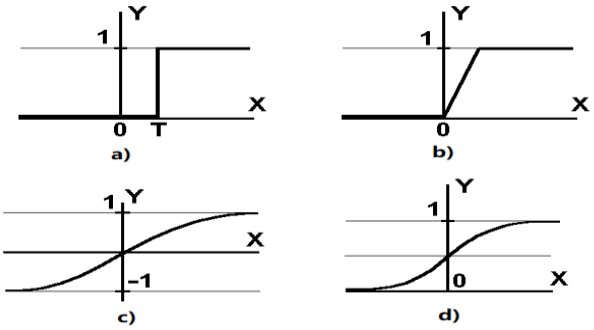
Pic. 1. Artificial neuron

The current state of a neuron is defined as the weighted sum of its inputs:

$$s = \sum_{i=1}^{n} x_i \cdot w_i \quad (1)$$

The output of a neuron is a function of its state:

$$y = f(s) \quad (2)$$



Pic. 2: a) Unit jump function; b) linear threshold (hysteresis); c) sigmoid – hyperbolic tangent; d) sigmoid – formula (3)

The non-linear function f is called an activation function and can take various forms, as shown in Pic 2. One of the most common is a non-linear function with saturation, the so-called logistic function or sigmoid (i.e., an S-shaped function) [2]:

$$f(x) = \frac{1}{1+e^{-\alpha x}}. \tag{3}$$

As α decreases, the sigmoid becomes flatter, in the limit at α=0 degenerating into a horizontal line at the level of 0.5, as α increases, the sigmoid approaches in appearance to the unit jump function with threshold T at the point x=0. From the expression for the sigmoid, it is obvious that the output value of the neuron lies in the range [0,1]. One of the valuable properties of the sigmoid function is a simple expression for its derivative, the application of which will be discussed later.

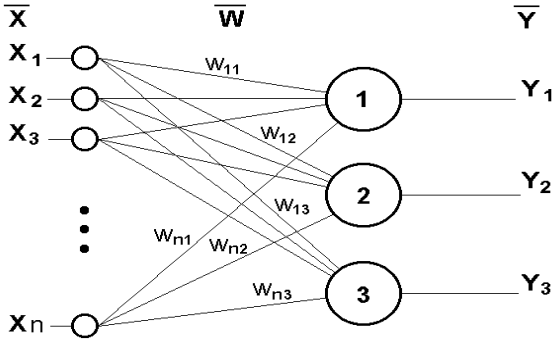
$$f'(x) = \alpha \cdot f(x) \cdot (1 - f(x)). \tag{4}$$

It should be noted that the sigmoid function is differentiable on the entire x-axis, which is used in some learning algorithms. In addition, it has the property of amplifying weak signals better than large ones, and prevents saturation from large signals, as they correspond to regions of arguments where the sigmoid has a gentle slope.

Returning to the common features inherent in all NNs, we note, secondly, the principle of parallel signal processing, which is achieved by combining a large number of neurons into so-called layers and connecting neurons of different layers in a certain way, and also, in some

configurations, neurons of one layer among themselves, and the processing of the interaction of all neurons is carried out in layers.

As an example of the simplest neural network, consider a three-neuron perceptron (Fig. 3), that is, a network whose neurons have an activation function in the form of a single jump*. Some signals arrive at n inputs, passing through the synapses to 3 neurons, forming the only layer of this NN and producing three output signals:



Pic. 3. Single layer perceptron

$$y_j = f\left[\sum_{i=1}^{n} x_i \cdot w_{ij}\right], \qquad j=1...3 \qquad (5)$$

It is obvious that all the weight coefficients of synapses of one layer of neurons can be reduced to the matrix W, in which each element wij specifies the value of the i-th synaptic connection of the j-th neuron. Thus, the process occurring in the NN can be written in matrix form:

$$Y=F(XW) \qquad (6)$$

where X and Y are the input and output signal vectors, respectively, F(V) is the activation function applied element by element to the components of the vector V.

Theoretically, the number of layers and the number of neurons in each layer can be arbitrary, but in fact it is limited by the resources of a computer or a specialized microcircuit, on which NN is usually implemented. The more complex the NN, the larger the tasks subject to it.

The choice of the structure of the NN is carried out in accordance with the features and complexity of the problem. To solve some individual types of problems, there are already optimal, to date, configurations, described, for example, in [2], [3], [4] and other publications listed at the end of the article. If the task cannot be reduced to any of the known types, the developer has to solve the difficult problem of synthesizing a new configuration. At the same time, he is guided by several fundamental principles: the network capabilities increase with an increase in the number of network cells, the density of connections between them and the number of selected layers (the effect of the number of layers on the network's ability to classify flat images is shown in Pic. 4 from [5]); the introduction of feedback, along with an increase in network capabilities, raises the question of the dynamic stability of the network; the complexity of the algorithms for the functioning of the network (including, for example, the introduction of several types of synapses – excitatory, inhibiting, etc.) also contributes to the strengthening of the power of the neural network. The question of the necessary and sufficient properties of the network for solving one kind of problem or another is a whole area of neurocomputer science. Since the problem of synthesizing NN strongly depends on the problem being solved, it is difficult to give general detailed recommendations. In most cases, the best option is obtained on the basis of intuitive selection.

It is obvious that the process of functioning of the neural network, that is, the essence of the actions that it is capable of performing, depends on the values of synaptic connections, therefore, having given a certain structure of the neural network that meets any task, the network developer must find the optimal values of all variable weight coefficients (some synaptic connections may be permanent).

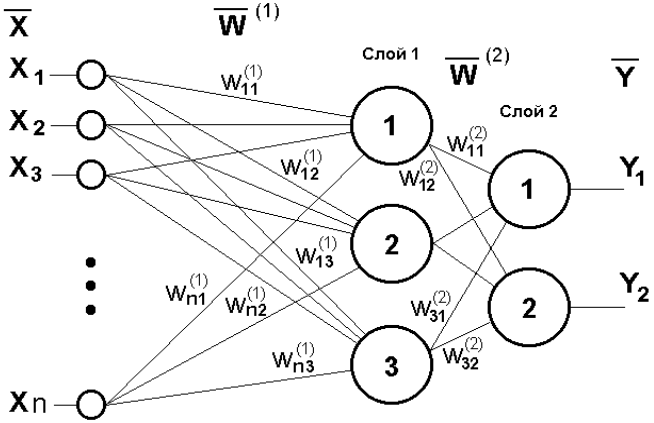
This stage is called NN training, and the ability of the network to solve the problems posed to it during operation depends on how well it is performed. At the training stage, in addition to the quality parameter of weight selection, training time plays an important role. As a rule, these two parameters are inversely related and have to be chosen on the basis of a compromise.

NN training can be conducted with or without a teacher. In the first case, the network is presented with the values of both input and desired output signals, and it adjusts the weights of its synaptic connections according to some internal algorithm. In the second case, the NN outputs are formed independently, and the weights are changed according to an algorithm that takes into account only the input signals and their derivatives.

There are a great many different learning algorithms, which, however, are divided into two large classes: deterministic and stochastic. In the first of them, the adjustment of the weights is a rigid sequence of actions; in the second, it is performed on the basis of actions that are subject to some random process.

Developing further the question of the possible classification of NN, it is important to note the existence of binary and analog networks. The first of them operate with binary signals, and the output of each neuron can take only two values: a logical zero ("inhibited" state) and a logical one ("excited" state). The perceptron considered above also belongs to this class of networks, since the outputs of its neurons, formed by the unit jump function, are either 0 or 1. In analog networks, the output values of neurons can take continuous values, which could take place after replacing the activation function of the perceptron neurons to the sigmoid.

Another classification divides neural networks into synchronous and asynchronous [3]. In the first case, only one neuron changes its state at a time. In the second, the state changes immediately for a whole group of neurons, as a rule, for the entire layer. Algorithmically, the course of time in the NN is set by iterative execution of the same type of actions on neurons. In what follows, only synchronous NN will be considered.



Pic. 4. Double layer perceptron

Networks can also be classified according to the number of layers. Figure 4 shows a two-layer perceptron obtained from the perceptron

in Pic. 3 by adding a second layer consisting of two neurons. Here it is appropriate to note the important role of the nonlinearity of the activation function, because if it did not have this property or was not included in the algorithm of each neuron, the result of the operation of any p-layer NN with weight matrices $W^{(i)}$, i=1,2,. ..p for each layer i would be reduced to multiplying the input signal vector X by the matrix

$$W^{(\Sigma)}=W^{(1)}\cdot W^{(2)}\cdot ...\cdot W^{(p)} \quad (7)$$

that is, in fact, such a p-layer NN is equivalent to a single-layer NN with a weight matrix of a single layer $W^{(\Sigma)}$:

$$Y=XW^{(\Sigma)} \quad (8)$$

Continuing the conversation about nonlinearity, it can be noted that it is sometimes introduced into synaptic connections. Most currently known neural networks use formula (1) to find the weighted sum of neuron inputs, however, in some applications of neural networks it is useful to introduce another notation, for example:

$$s=\sum_{i=1}^{n} x_i^2 \cdot w_i \quad (9)$$

or even:

$$s=\sum_{i=1}^{n} x_i \cdot x_{((i+1) \bmod n)} \cdot w_i \quad . \quad (10)$$

The question is that the NN developer clearly understands why he is doing this, what valuable properties he thereby additionally endows the neuron, and which it deprives. The introduction of this kind of nonlinearity, generally speaking, increases the computational power of the network, that is, it makes it possible to construct an NN from a smaller number of neurons with "nonlinear" synapses that performs the work of a conventional NN with a large number of standard neurons and a more complex configuration [4].

The variety of existing NN structures allows finding other criteria for their classification, but they are beyond the scope of this article.

Now consider one nuance, deliberately omitted earlier. It can be seen from the figure of the unit jump function that the threshold value T, in the general case, can take an arbitrary value. Moreover, it must take some arbitrary value, unknown in advance, which is selected at the training stage along with weight coefficients. The same applies to the center point of the

sigmoid curve, which can shift to the right or left along the x-axis, as well as to all other activation functions. This, however, is not reflected in formula (1), which should have looked like this:

$$s = \sum_{i=1}^{n} x_i \cdot w_i - T \quad . \tag{11}$$

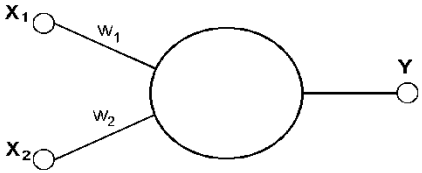
The fact is that such a bias is usually introduced by adding another input to the layer of neurons, which excites an additional synapse of each of the neurons, the value of which is always equal to 1. Let's assign the number 0 to this input. Then

$$s = \sum_{i=0}^{n} x_i \cdot w_i \quad , \tag{12}$$

where $w_0 = -T$, $x_0 = 1$.

Obviously, the difference between formulas (1) and (12) is only in the method of numbering the inputs. Of all the activation functions depicted in Pic. 2, one stands out in particular. This is a hyperbolic tangent whose dependence is symmetrical about the X axis and lies in the range [-1,1]. Looking ahead, let's say that the choice of the range of possible values of neuron outputs largely depends on the specific type of neural network and is a matter of implementation, since manipulations with it affect various network performance indicators, often without changing the overall logic of its operation. An example illustrating this aspect will be presented after moving from a general description to specific types of NN.

What tasks can the NN solve? Roughly speaking, the work of all networks is reduced to the classification (generalization) of input signals belonging to the n-dimensional hyperspace, according to a certain number of classes.



Pic. 5. Single neuron perceptron

$$\sum_{i=1}^{n} x_i \cdot w_{ik} = T_k \text{ , k=1...m} \quad (13)$$

Each resulting scope is the scope of a separate class. The number of such classes for one perceptron-type NN does not exceed 2m, where m is the number of network outputs. However, not all of them can be separable by a given NN.

For example, a single-layer perceptron consisting of one neuron with two inputs, shown in Figure 5, is not able to divide a plane (two-dimensional hyperspace) into two half-planes in such a way as to classify input signals into classes A and B (see Table 1).

Table 1

$x_1 x_2$	0	1
0	A	B
1	B	A

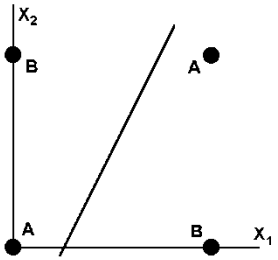
The network equation for this case:

$$x_1 \cdot w_1 + x_2 \cdot w_2 = T \quad (14)$$

is the equation of a straight line (one-dimensional hyperplane), which under no circumstances can divide the plane so that the points from the set of input signals belonging to different classes are on opposite sides of the straight line (see Pic. 6).

If you look closely at Table 1, you can see that this categorization implements a logical XOR function for input signals. The impossibility of implementing this function by a single-layer perceptron is called the XOR problem.

Functions that are not implemented by a single-layer network are called linearly inseparable [2]. The solution of problems that fall under this restriction is to use 2 or more layer networks or networks with non-linear synapses, however, even then there is a possibility that the correct division of some input signals into classes is impossible.



Pic. 6. Visual representation of the work of the NN from Pic. 5

Finally, we can consider the issue of training the NN in more detail, for a start, using the example of the perceptron in Pic. 3.

Consider a supervised learning algorithm [2] [4].

1. Initialize the elements of the weight matrix (usually with small random values).

2. Submit to the inputs one of the input vectors that the network must learn to distinguish, and calculate its output.

3. If the output is correct, go to step 4.

Otherwise, calculate the difference between the ideal and the obtained output values:

$$\delta = Y_I - Y.$$

Modify the weights according to the formula:

$$w_{ij}(t+1) = w_{ij}(t) + \nu \cdot \delta \cdot x_i.$$

where t and t+1 are the numbers of the current and next iterations, respectively; ν – learning rate coefficient, 0<νJ1; i – entry number; j is the number of the neuron in the layer.

Obviously, if $Y_I > Y$, the weighting factors will be increased and thereby reduce the error. Otherwise, they will be reduced, and Y will also decrease, approaching Y_I.

4. Loop from step 2 until the network stops making mistakes.

At the second step, at different iterations, all possible input vectors are presented one by one in a random order. Unfortunately, there is no way to predetermine the number of iterations that will need to be performed, and in some cases even guarantee complete success. This issue will be touched upon later.

At the end of this introductory article, I would like to note that further consideration of the NN will mainly gravitate towards such applications as pattern recognition, their classification, and, to a small extent, information compression. More details about these and other applications and the NN structures that implement them can be found in the journals Neural Computation, Neural Computing and Applications, Neural Networks, IEEE Transactions on Neural Networks, IEEE Transactions on System, Man, and Cybernetics, and others.

References:

1. Е. Монахова, "Нейрохирурги" с Ордынки, PC Week/RE. 1995. № 9.
2. Ф.Уоссермен, Нейрокомпьютерная техника. М., Мир, 1992.
3. Итоги науки и техники: физические и математические модели нейронных сетей. Том 1. М., изд. ВИНИТИ, 1990.
4. Artificial Neural Networks: Concepts and Theory, IEEE Computer Society Press, 1992.
5. Richard P. Lippmann, An Introduction to Computing withNeural Nets, IEEE Acoustics, Speech, and Signal ProcessingMagazine, April 1987.