NEURAL NETWORK OPERATING PECULIARITIES

Oleh VARHACH[1], Kseniia KUGAI[2*]

[1] Kyiv National University of Technologies and Design, Mechatronics and Computer Technologies Faculty, Computer Science and Technologies Department, BITsk-19, Kyiv, Ukraine
[2] Kyiv National University of Technologies and Design, Senior Teacher of Foreign Languages Department

*Corresponding author: Kseniia Kugai, sketch_k2008@ukr.net

Abstract. *The article analyzes operating peculiarities of the standard neural network for a multilayer perceptron. The network structure and the principle of inverse error propagation algorithm operation are described.*
The article deals with the notions of artificial neural networks and deep structured learning. Neural networks are the basis of a new and interesting area – deep structured learning.
The work covers basic concepts, as well as some code and math, which can help understand and build simple neural networks.

Keywords: *algorithm, deep structured learning, neuron, artificial, multilayer perceptron.*

Introduction and problem statement

Deep learning is the field of machine learning that has helped us make a big breakthrough in many ways today, from playing Go and Poker with live players to drones. But, first of all, deep structured learning requires knowledge of neural networks operation.

Artificial neural networks (ANNs) are software implementations of neural structures in our brain. One should know that the brain contains neurons, which are a kind of organic switches. They can change the type of signals transmitted depending on the electrical or chemical signals transmitted in them. A neural network in the human brain is a huge interconnected system of neurons, where a signal transmitted by one neuron can be transmitted to thousands of other neurons. Learning occurs through the reactivation of some neural connections. This increases the probability of outputting the desired result with the appropriate input information (signals). This type of learning uses feedback with the right result; the neural connections that output it become denser [1].

Artificial neural networks mimic the behavior of the brain in a simpler form. They can be taught in controlled and uncontrolled ways.

In controlled ANNs, the network is trained by transmitting relevant input information and kind of output information. For example, spam filter in an e-mail box: the input information can be a list of words that are usually in spam messages, and the output information can be a classification for the corresponding message. This type of training adds weight to ANN connections.

Uncontrolled learning in an artificial neural network tries to force ANN to understand the structure of the transmitted input information “independently”.

Experimental part

Let us assume that we want to teach a computer to recognize handwritten numbers. You can use classical mathematical methods to write a program that can identify specific features that distinguish one number from another. For example, “eight” has two circles, and “seven” has two long straight lines, and so on. But the program would have to manually identify these features and describe them, and this is a huge amount of work. Neural networks are doing an excellent job with such tasks today, because they are able to find and identify these features on their own.

For example, let us take a classical structure neural network called a multilayer perceptron Fig. 1.

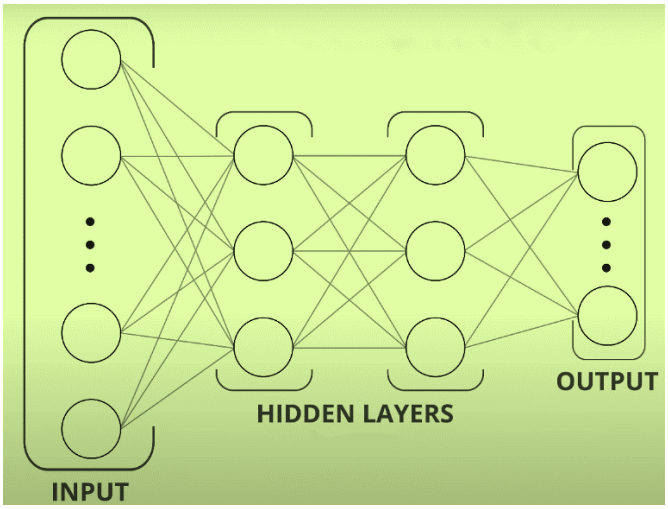

Figure 1. Scheme of a multilayer perceptron

A neural network consists of neurons, and each neuron is a cell that stores a limited range of values. In our case these will be values from 0 to 1. A set of values is received at the input of each neuron, and at the output it gives only one. This neural network is called multilayer because the neurons in it are organized into columns, and each column is a separate layer. In our case there are 4 layers.

The first layer is called “input”, in fact, it just receives input data. For example, if we want to recognize a picture with a figure of 28 by 28 pixels, we need 784 neurons in the first layer of our neural network by the number of pixels in the picture. Since the neural network can only store values from 0 to 1, we will encode the brightness of each pixel in this range of values. The next two layers are called “hidden layers”. The number of neurons in the hidden layers can be whatever you want; it is selected by trial and error. These are the layers responsible for identifying specific features. The value from the input layer goes to the hidden layers. Mathematical calculations take place there and after transformation they are sent to the layer called “output”. And the neuron of the output layer, in which the highest value is found, is considered to be the response [2-4].

Speaking about our neural network, we recognize numbers and in the output layer we have 10 neurons, denoting an answer from 0 to 9. But what kind of data is transmitted over the layers and what kind of mathematical calculations take place inside? Let us consider this using one of the neurons in the second layer as an example. This neuron, as well as other hidden layers, receives the sum of all the values of input layer neurons. The task of the neurons of the second layer is to find some features, for example, this neuron could search for the horizontal line of the upper part of number “seven”.

If we acted on classical algorithm logic, we could assign different coefficients to different areas. For example, we assume that there should be bright pixels in the upper part of the image, a horizontal stick at “seven”. For this area, we can set higher coefficients, and for other areas, lower ones. Such coefficients in neural networks are usually called weights, and in formulas they are denoted by the letter “w” [2].

Now, if we multiply the input brightness values by the weights, we will understand whether the stick was in this area or not. If a sign is found in the neuron, a larger number will be written. And if there was no sign, the number would be small. But in order to activate the neuron, we need to supply a sufficiently high number there, above some threshold value. Otherwise nothing else will transfer. We know that a neuron can contain a value from 0 to 1, while the input data can have much higher values. Not only do we sum the values from the first layer, we also multiply them by weights. So we need to normalize the value obtained, for example, using a function such as sigmoid or ReLU Fig. 2 [5].

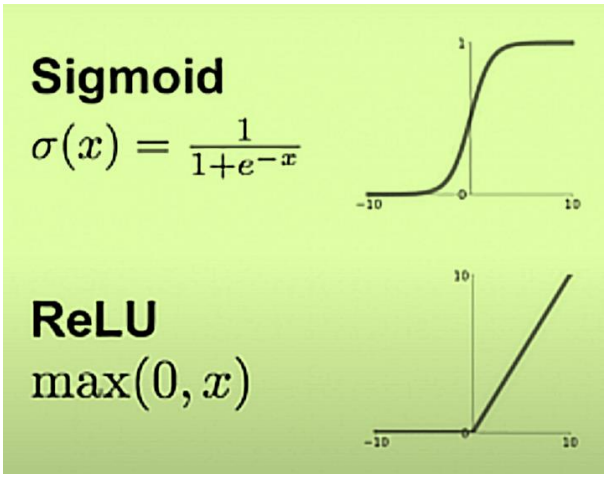

Figure 2. Formula and graph of the sigmoid and ReLU

But if in the original pictures there are some noise, dots and dashes, the rest of this noise needs to be somehow cut off. For this purpose the coefficient of shift "bias" is entered into the formula. It is designated by the letter "b".

For example, if the bias negative neuron should be used less frequently, this function is called "activation function". All these weights and biases for each neuron are adjusted separately, and even in such a seemingly simple neural network, there are about 13,000 of them. All this is very difficult to do manually. In order to add the right weights, we just give the neural network arbitrary values of weights and displacements. As a result we obviously get completely random answers at the output. But we have the advantage of knowing the correct answers, which means that in each specific case, we can indicate to the neural network how wrong it was.

Here the algorithm of back propagation of an error comes into play. It took as long as 25 years to develop this method.

Results and discussion

Suppose we loaded number "two" into the neural network. If the neural network worked perfectly in the output neuron responsible for recognizing "two", the maximum value would be equal to 1, and in the rest of the neurons there would be zeros. This means that the neural network is 100% sure that this "two", and nothing else, and we got other values. But since we know the correct answer, we can subtract the correct ones from the wrong answers and calculate how much the neural network was wrong in each case. And then knowing the degree of error, we can adjust the weights and bias for each neuron in proportion to how much they contributed to the total error [3].

Naturally, having performed such an operation once, we will not be able to achieve the correct values at the output. But with each attempt the total error will decrease and only after hundreds of thousands of direct propagation error cycles the reverse neural network will be able to pick up the optimal weights and displacements.

Conclusions

This is how neural networks and machine learning work in classification problems. This is the simplest example, there are also many other neural network architectures: neural networks with and without a teacher; neural networks that teach each other and compete with each other; neural networks that self-adjust their structure in the learning process, just as it happens in the human brain. But each of them carries one idea, to facilitate and accelerate human work and open new opportunities in solving various problems. Neural networks are the whole world of extremely interesting knowledge.

References

1. Neironni merezhi – shliakh do hlybynnoho navchannia [online]. [viewed on 13.01.2021]. Retrieved from: https://codeguida.com/post/739
2. Iskusstvennyiy intellekt: chto mogut neyronnyie seti i kak oni izmenyat nashu zhizn? [online]. [viewed on 12.01.2021]. Retrieved from: https://www.kommersant.ru/doc/3495930
3. Metod zvorotnoho poshyrennia pomylky [online]. [viewed on 13.01.2021]. Retrieved from: https://znaimo.com.ua/Метод_зворотного_поширення_помилки
4. Nechuvstvitelnyie k vesam neyronnyie seti (WANN) [online]. [viewed on 12.01.2021]. Retrieved from: https://m.habr.com/ru/post/465369/
5. Wasserman, F. *Neyrokompyuternaya tehnika: Teoriya i praktika* [Neurocomputer Engineering: Theory and Practice]. Moscow: Mir, 1992.