UDC 004.838.5

# PROBLEMATICS OF THE CONTROL SYSTEM AND THE POSSIBILITY OF THEIR SOLUTION WITH NEURAL NETWORKS

*VASHKEVICH MAKSIM, DZMITRY PIATKIN*
Polotsk State University, Belarus

*This article describes neural networks, their creation and possible ways of using. Also, the possibility of using a neural network is described in more detail.*

In order to automate a system, it is necessary to remove a person either at some stage or from the entire system. In order to achieve this, you can use different possibilities. However, in this paper, we will consider the possibility of automating a system, or any process, with the help of neural networks.

To understand why neural networks were chosen to solve automation tasks it's essential to understand that they exist, how they function now, why they are the most suitable option for solving automation problems, and where neural networks are still possible.

First of all, neural networks as a mathematical model, as well as their software or hardware implementation, are built on the principle of the organization and functioning of biological neural networks - networks of nerve cells of a living organism. This concept arose when studying the processes occurring in the brain, and when trying to simulate these processes. After the development of learning algorithms, the resulting models began to be used for practical purposes: in tasks forecasting, for pattern recognition, and also in control tasks. [1] In this description, it can be seen that this type of software has evolved as an attempt to create a system that can act in the likeness of the human brain and can also be decision based on the incoming data.

Now neural networks are used for pattern recognition, as well as for classification. Both of these tasks, if solved, together with a small upgrade, can serve as the basis for automation. As an example, consider the 2 systems in which the computer has already replaced a person. One of such systems is a face recognition or fingerprint recognition system. Previously, in order to find a person by fingerprint, a team of specialists was used, who alternately compared the fingerprints from the file cabinet with the necessary print sent for comparison. As a result, it took a huge amount of man-hours to compare and reveal to whom this or that fingerprint belonged. Now a special joint system from the Gabor filter, the five-level Daubechies wavelet transform and the multilayer neural network are used to perform this task. The use of these principles allows you to recognize and classify fingerprints. [2] The general scheme of the work of classification of a fingerprint can be viewed in Figure 1.

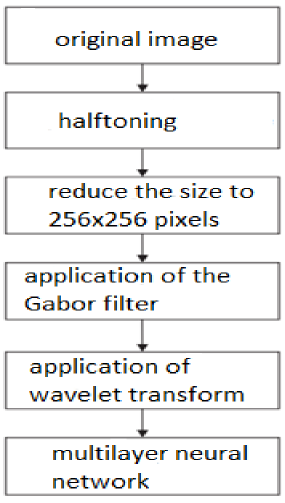

original image
halftoning
reduce the size to 256x256 pixels
application of the Gabor filter
application of wavelet transform
multilayer neural network

Fig. 1. General scheme for the classification of fingerprints

As described above, automation of the system with software is an attempt to remove the human being from the system or some part of the system. So this is an attempt to delegate the decision making process to the machine. Since a person makes decisions with the brain, it is logical to assume that to solve the problem of automating a certain process, it is necessary to use something that will act as the human brain. And this means that to solve such a task the neural network that will act as a human being is best suited, but unlike a human, it will not make human mistakes. So, for example, the percentage of error of any neural network can be known in advance. And if the percentage error is not satisfactory, the neural network can be modified to meet the requirements. Thanks to the fact that the neural network can replace a person at the proper level, neural networks have automated and replaced a person on a large number of industries.

Since the neural network should ideally act as the human brain and make decisions, it is worth creating a complex system that will act like a human being. A typical multi-variable system is driving a vehicle or controlling an unmanned vehicle using a control panel. In this report, we will consider the possibility of automating the motion of unmanned objects using the neural network.

At the moment, there are already systems that allow unmanned objects to be managed independently. Such systems can include robots vacuum cleaners and squares for the delivery of goods. However, it should be noted that both these systems are fairly simple and do not take into account more complex options. So, robots vacuum cleaners just move around the room and avoid obstacles if they rest on them, and the squares move in a straight line, simply bypassing the largest buildings. This means that both systems take into account a small number of obstacles on the way. This problem can be identified if we run a robot vacuum cleaner in a labyrinth or run a squarchop of delivery in the forest and do not let it rise above tree trunks. So to automate such a thing as managing an unmanned vehicle in a complex environment, you will have to develop a system that can make a decision with a lot of obstacles. To make the system look like a person, the system should see obstacles just before itself (so the system can act as well as a person choosing from the options that are in front of it at the moment). Suppose that we want our unmanned vehicle to be controlled by itself, for this we need to do the following: first, we create a neural network that will control the unmanned vehicle and avoid obstacles. We achieve this by selecting appropriate inputs / outputs and careful training of the neural network. We transmit the network distance to the nearest obstacles around the apparatus, imitating the sight of a person. At the output, we get acceleration and turns for the unmanned vehicle. We also need to train the network on a variety of I / O strategies. The drone moves bypassing obstacles minimum for motion, the rest can be added for the system depending on what the system needs to do.

The network can be compared to a "function". There are many entrances. An entrance to the entrance is an obstacle array. These data are processed by the neuronet as a function and the response of the neurons is an exit from the function. The function f (x) = y converts the value of x (one dimension) to y (one dimension). We use a neural network back propagation.

This neural network performs the work most quickly with several inputs. Since we have to work with several inputs and outputs, this system is the most suitable. When a neural network consists of only a few neurons, we can calculate the weights necessary to obtain an acceptable result. But as the number of neurons increases, the complexity of computations increases. A network of backward propagation can be trained to establish the necessary weights. We simply need to provide the desired results with their corresponding inputs. After training, the neural network will respond and produce a result close to the desirable result when submitting a known result and "guessing" the correct answer at any input not corresponding to the instructor. [3]

To train the neural network, we will use the backward propagation of the error, as well as training with the teacher. For training a neural network with the teacher, it is necessary to input the possible combinations of the inputs of the outputs. After training, the network will be able to make decisions according to what data it has received. This type of training was used in connection with the fact that it gives the smallest error in the calculations when learning a neural network.

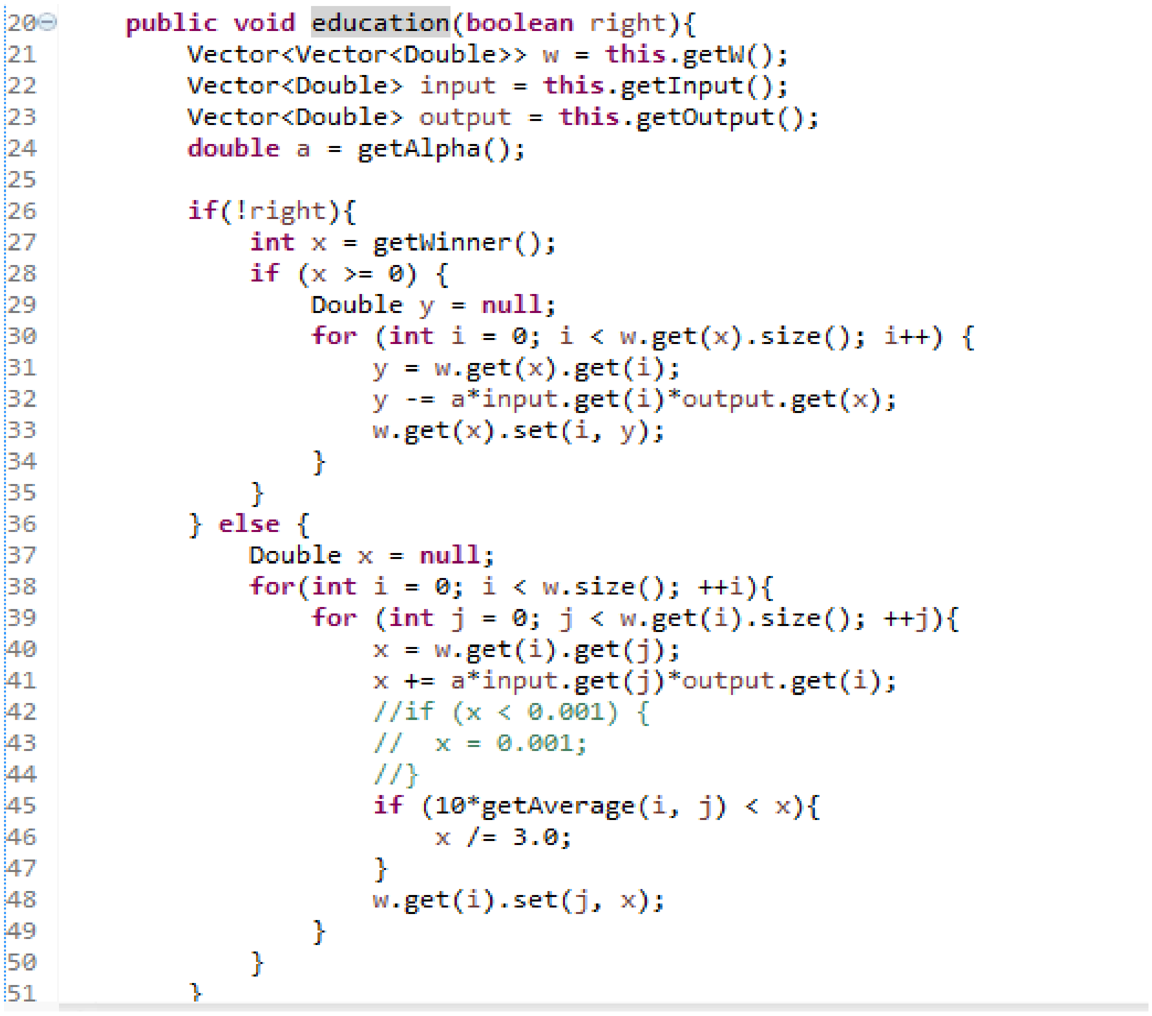
Now that we've looked at how the "brain" works, we need to understand how to determine the inputs and outputs of the neural network. The neural network does not by itself do anything if we give it information and do not give the network response to the vehicle controller.

An obstacle matrix is fed into the neural network, this matrix contains the notation whether there is an obstacle in front of the movement or not. The matrix must contain elements, each of which must be zero or one, zero means that there are no obstacles in this cell and you can move there, if there is one in the cell, it means that there is an obstacle there and it is impossible to move there. Each cell represents the area in front of the drone, these areas are divided into N parts and numbered from the top left edge to the left to the right, top to bottom [4].

On the output from the neural network we can enter the choice: either the angles of rotation of the unmanned vehicle or the number of the cell where it is necessary to move.

From all that has been said, we can conclude that even such a complex action as driving a vehicle in a complex environment can be shifted to managing by neural networks, which will help to remove the human factor and human errors. In addition, it will help in economic terms, since neural networks do not require constant cash injections and are limited only to purchase and, in case of change in some factors, modernization.

To implement the neural network, the network was divided into 2 neural networks, each of them was trained using its own algorithm. To train the first neural network, Hebb training is used. Since the first neural network uses training without a teacher (Hebb method), some shortcomings have been identified in the development process. The main disadvantage is that the neural network is not always properly trained. In order to remove this drawback, the network was trained. After the tests, the appropriate weights for the neural network were chosen and they remained as standard. Due to this, errors related to incorrect learning of the neural network were eliminated. The training itself can be seen in Figure 2.

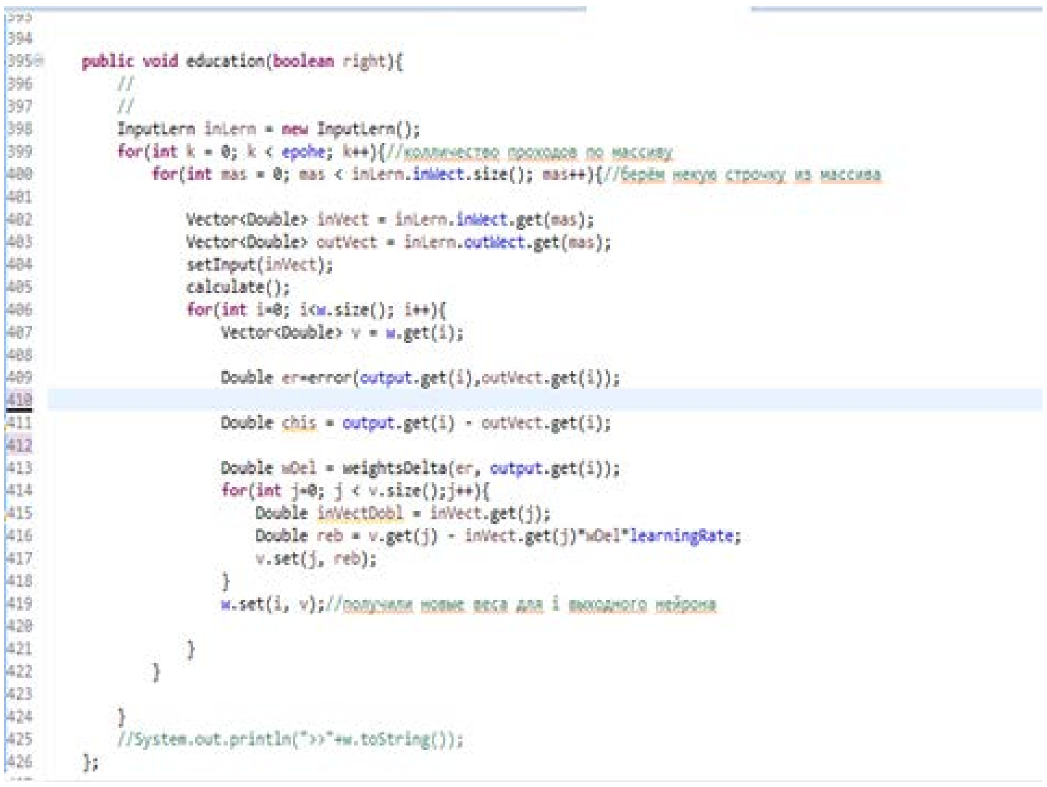


```
    public void education(boolean right){
        Vector<Vector<Double>> w = this.getW();
        Vector<Double> input = this.getInput();
        Vector<Double> output = this.getOutput();
        double a = getAlpha();

        if(!right){
            int x = getWinner();
            if (x >= 0) {
                Double y = null;
                for (int i = 0; i < w.get(x).size(); i++) {
                    y = w.get(x).get(i);
                    y -= a*input.get(i)*output.get(x);
                    w.get(x).set(i, y);
                }
            }
        } else {
            Double x = null;
            for(int i = 0; i < w.size(); ++i){
                for (int j = 0; j < w.get(i).size(); ++j){
                    x = w.get(i).get(j);
                    x += a*input.get(j)*output.get(i);
                    //if (x < 0.001) {
                    //   x = 0.001;
                    //}
                    if (10*getAverage(i, j) < x){
                        x /= 3.0;
                    }
                    w.get(i).set(j, x);
                }
            }
        }
```

Fig. 2. Training of the first neural network

The second neural network is trained using the method of back propagation of the error; the implementation of this method can be seen in Figure 3.



```
    public void education(boolean right){
        //
        //
        InputLern inLern = new InputLern();
        for(int k = 0; k < epohe; k++){//количество проходов по массиву
            for(int mas = 0; mas < inLern.inWect.size(); mas++){//берём некую строчку из массива

                Vector<Double> inVect = inLern.inWect.get(mas);
                Vector<Double> outVect = inLern.outWect.get(mas);
                setInput(inVect);
                calculate();
                for(int i=0; i<w.size(); i++){
                    Vector<Double> v = w.get(i);

                    Double er=error(output.get(i),outVect.get(i));

                    Double chis = output.get(i) - outVect.get(i);

                    Double wDel = weightsDelta(er, output.get(i));
                    for(int j=0; j < v.size();j++){
                        Double inVectDobl = inVect.get(j);
                        Double reb = v.get(j) - inVect.get(j)*wDel*learningRate;
                        v.set(j, reb);
                    }
                    w.set(i, v);//получили новые веса для i выходного нейрона

                }
            }

        }
        //System.out.println(">>"+w.toString());
    };
```

Fig. 3. Realization of training of the second neural network

Thus, the work examined the possibilities of a library using neural networks. Various ways of using the library were considered. In addition, a concrete example of using the library was considered in detail to simplify the performance of some tasks.

REFERENCES

1. Description of what a neural network. [Electronic resource] – Mode of access: https://en.wikipedia.org/wiki/Artificial_neural_network. – Date of access: 14.11.2017.
2. Using Neural Networks [Electronic resource] – Mode of access: https://www.slideshare.net/dhan1989/face-recognization-using-artificial-nerual-network?next_slideshow=1. – Date of access: 14.11.2017.
3. Description of neural networks [Electronic resource]. – Mode of access: https://ru.wikipedia.org/wiki/. – Date of access: 27.09.2017.
4. Movement in the plane. [Electronic resource] – Mode of access: http://neuralnetworksanddeeplearning.com/. – Date of access: 27.09.2017.