Adv Syst Sci Appl 2022; **04**; 212-223
Published online at http://ijassa.ipu.ru.

Computer Program to Automate the Determination of the Size of Sanitary Protection Zones (SPZS) and Surveillance Zones for the Storage of Sources of Ionizing Radiation in places of Burial / Storage of Radioactive Waste

Petr Istratov[1], Yury Minaev[1], Elena Zarubina[1], Petr Zolotariov[1]

[1)] *Medical University Reaviz, Moscow, Russia*

Abstract: The rapid development of nuclear power and industry, has led to the need to address the issues of waste disposal, which are sources of ionizing radiation and can be hazardous to human health. In this regard, the development of automated systems for preliminary assessment of the place of future disposal of radioactive waste (RW) and the width of the sanitary protection zone (SPZ), to avoid falling outside residential areas, is an urgent task. The purpose of this study was to develop algorithms for solving the task of processing and analyzing information for modeling the operation of the radioactive waste disposal system, as well as to test the performance of these algorithms in practice. On the basis of artificial intelligence (neural network) the "Computer program for automation of determination of dimensions of sanitary-protective zones (SPZ) and surveillance zones for storage of sources of ionizing radiation in places of burial/storage of radioactive wastes" was developed. The main advantage of the developed neural network is that it can identify non-obvious attributes for determining the size of sanitary protection zones (SPZ). These algorithms do not rely on the existing knowledge about the influence of the analyzed indicators, which makes them objective and allows to use them to create models of the state of storage of radiation sources.

Keywords: neural networks, algorithm, sanitary protection zone.

1. INTRODUCTION

Rapid development of nuclear power and industry has led to the necessity of solving the issues of waste disposal, which are sources of ionizing radiation and may pose a danger to human health.

All issues related to the organization of measures to protect the population living in the vicinity of such facilities are strictly regulated, however, when choosing a place for disposal of radioactive waste one always has to take into account, in addition to the class of waste, the combined effect of various factors both natural and climatic (wind rose, amount of precipitation, soil nature, natural radioactive background of the area, presence or absence of forests, their nature and much more) and anthropogenic nature (for example, presence of industrial waste). In this connection, development of automated systems for preliminary assessment of the place of future disposal of radioactive waste (RW) and the width of the sanitary protection zone (SPZ), in order to avoid getting outside the zones of residential development, is an urgent task.

The purpose of this study was to develop algorithms for solving the problem of information processing and analysis to simulate the operation of the radioactive waste disposal system, as well as to test the performance of these algorithms in practice.

In general case the problem to be solved refers to the problem of classification. This is the main and very extensive group of medical and biological problems. The answer in them is a class - the choice of one variant from a predetermined set of variants. For our problem,

classification is defined as a binary problem (elementary classification) - in this case, the set of possible answers consists of two options (classes), with the first option determining the size of the obtained sanitary protection zone as sufficient, and the second option as not adequate enough to solve the problem.

2. OBJECTIVES

The scientific goal of this work consisted of solving the following main tasks:

1. Choose the form of AI implementation for the developed program;
2. Analyze and select the most significant indicators for the system development;
3. Create an expert system based on the selected indicators;
4. Test the system operation under conditions of real estimation of the SWD width in the area of the RAW disposal site.

3. METHODS

Several stages of development of "Computer program for automation of determining the size of sanitary protection zones (SPZ) and surveillance zones for storage of sources of ionizing radiation in places of burial / storage of radioactive waste" can be distinguished in creation of a self-training neural network, some of which coincide with the stages of creation of traditional systems.

Problem formulation: the same as for the traditional systems plus the choice of the optimal neural network structure and training methods (for most tasks the structure and methods are standard).

1. Collection of training data.

A set of examples for training the network, each of which represents an array of input data and its corresponding response known in advance.

2. Creating and training a neural network.

If the task is within the standard scheme (in most cases), you don't need to do any statistical calculations, but programming work, too. If the task is non-standard, you need to adapt the structure of a neural network and the method of estimation calculation in training. Training of a neural network in most standard cases is an automatic process, which only after its completion requires a specialist to evaluate the results. Of course, it may often require correction, creation of additional networks with other parameters, etc., but it is always possible to evaluate the system performance at any stage of training by testing a control sample. Developing methodology of neural network expert systems, we proceeded from the possibility to develop the most individualized (designed for one particular user-specialist) systems by this specialist. Of course, nothing prevents to combine in one system individual experience of several experts. The absence of "mathematical" stages realizes such possibilities. The subject specialist is able to set the task himself; moreover, no one but him can do it better. The subject specialist must also collect the material. Schemes of problem setting, ways of data representation and ways of response production by a neural network are designed in such a way that most tasks in many fields fit into these standard schemes. So with well-designed neural network software tools and neural network documentation, most specialists are able to develop not very complex neural network applications on their own.

3. Interface creation.

The same as for traditional expert systems.

4. Debugging and testing.

This stage mainly includes debugging of the program, since testing is often carried out during the training of networks.

5. Post-training.

This stage is typical only for learning systems. When creating neuroexpert programs it is quite seldom possible to collect enough data for good network training at once. So when creating a neural network researchers determine the best parameters of networks and do the initial training. Subsequently users train the system in conditions of real work and real data, transferring the experience to it. Moreover, the fundamental difference of methodology of neural network systems creation from traditional ones is exactly, that system is never created at once ready and never completely finished, continuing to accumulate experience in the process of operation.

Let's look at the structure and functioning of an individual neuron. Each connection from neuron to neuron is called a synapse. Fig. 1 shows a neuron with a group of synapses connecting the neuron either to other neurons or to the outside world. To consider how a neuron works, it does not matter whether the signal comes to the neuron from the outside world or from another neuron, and it does not matter where the signal is sent from the neuron. In full-connected networks the output signal is sent to all other neurons.

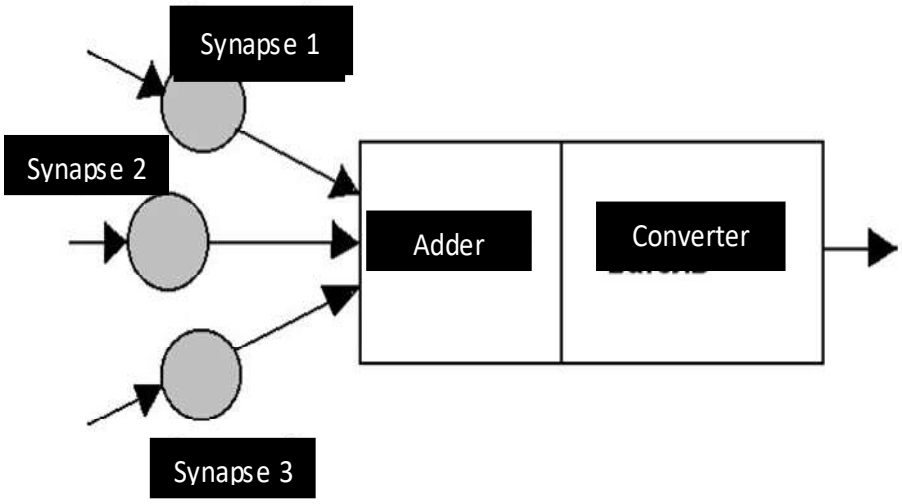

Fig. 1. Diagram of a neuron

A neuron consists of two functional units: the input adder and the neuron itself, or the transducer.

A neuron functions as follows: at the current moment signals from other neurons and/or from the external world are sent to it via input synapses (there are 3 on the figure). Each synapse has a parameter called synapse weight, which represents some number. A signal passing through a synapse is multiplied by the weight of that synapse. Depending on the weight, the signal can be amplified (weight modulus > 1) or attenuated (weight modulus < 1) in amplitude. The signals from all the synapses leading to a given neuron are received by the adder.

The summator sums all incoming signals and sends one number, the resulting sum, to the neuron itself (transducer). The value of this number will depend both on the values of initial signals and synapse weights. The neuron, which receives this number, transforms it according to its function resulting in another number, and sends it to all other neurons via the "axon" through the corresponding synapses. Subsequent neurons perform the same operations with the received signals, the only difference being that, firstly, the weights of their synapses may already be different, and secondly, other neurons may have another type of the transformation function. In the neural networks we construct, all neurons have the same function. This function, called the characteristic function, has the form:

$$f(X) = X/(C+|X|), \tag{1.0}$$

where X is the signal coming from the adder, C is a constant called the neuron characteristic. Experimentally we got that the optimal range of the characteristic for solving

the vast majority of problems is from 0.1 to 0.8. Plots of the characteristic function for both cases are shown in Fig. 2. The choice of such a function is due to the fact that it is smooth and continuous over the entire range of variables X, the range of values is always limited.

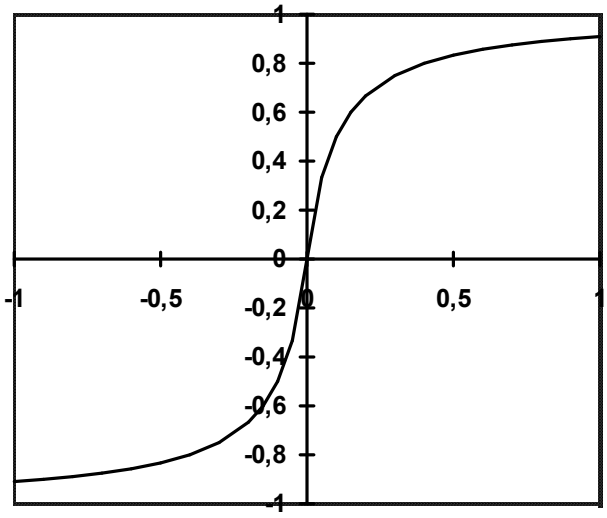

Fig. 2. Diagram of the characteristic function

Training the back propagation network includes the following operations [24]:

1. Select the next training pair from the training set; feed the input vector to the input of the network.
2. Compute the output of the network.
3. Calculate the difference between the output of the network and the required output (target vector of the training pair).
4. Adjust the weights of the network to minimize the error.
5. Repeat steps 1 to 4 for each vector of the training set, until the error on the whole set reaches an acceptable level.

The operations in steps 1 and 2 are similar to those in the already trained network, i.e., you feed the input vector and calculate the resulting output. The computation is performed in layers. In Fig. 3.3 first the outputs of neurons of layer j are computed, then they are used as inputs of layer k, the outputs of neurons of layer k are computed, which form the output vector of the network.

In step 3, each of the outputs of the network, which in Fig. 3 are labeled OUT, is subtracted from the corresponding component of the target vector to get an error. This error is used in step 4 to correct the weights of the network, with the sign and magnitude of the weight changes determined by the learning algorithm (see below).

After a sufficient number of repetitions of these four steps, the difference between the actual outputs and the target outputs should decrease to an acceptable value, and the network is said to be trained. Now the network is used for recognition and the weights are unchanged.

Steps 1 and 2 can be looked at as a "forward pass" as the signal propagates through the network from input to output. Steps 3, 4 constitute the "reverse pass", here the calculated error signal propagates back through the network and is used to adjust the weights. These two passes will now be detailed and expressed in a more mathematical form.

Forward passage. Steps 1 and 2 can be expressed in vector form as follows: input vector X is fed and the output is vector Y. The vector pair input-target X and T is taken from the training set. Calculations are performed on vector X to get output vector Y.

As we have seen, calculations in multilayer networks are performed layer by layer, starting from the layer closest to the input. The NET value of each neuron of the first layer is calculated as a weighted sum of the neuron's inputs. Then the activation function F "compresses" the NET and gives the OUT value for each neuron in that layer. When the

output set of a layer is obtained, it is the input set for the next layer. The process is repeated layer by layer until the final set of outputs of the network is obtained.

This process can be expressed in condensed form using vector notation. The weights between neurons can be thought of as a matrix W. For example, the weight from neuron 8 in layer 2 to neuron 5 in layer 3 is denoted by w8,5. Then the NET vector of layer N can be expressed not as the sum of products, but as the product of X and W. In vector notation, N = XW. Component application of the function F to the NET-vector N yields the output vector O. Thus, for this layer the computational process is described by the following expression:

$$O = F(XW) \tag{2.1}$$

The output vector of one layer is the input vector for the next layer, so calculating the outputs of the last layer requires applying equation (2.3) to each layer from the network input to its output.

Reverse pass. Adjusting the weights of the output layer. Since each neuron in the output layer has a target value, weights can easily be adjusted using a modified delta rule. The inner layers are called "hidden layers", for their outputs there are no target values for comparison. Therefore, the training becomes more complicated.

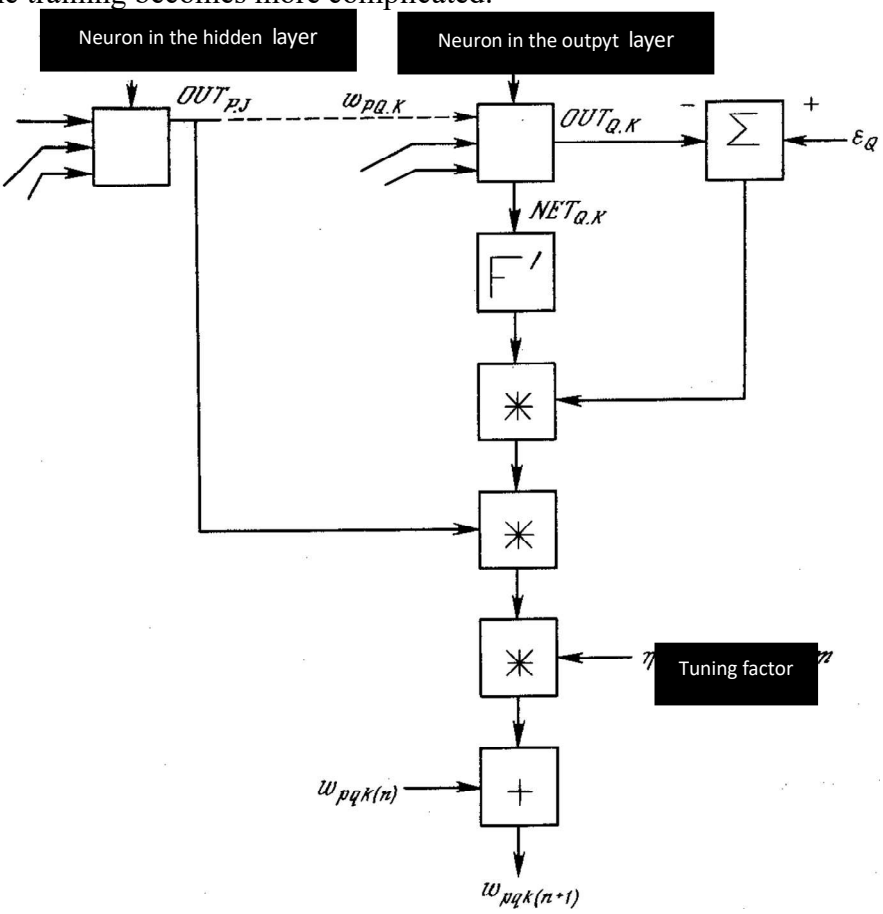

Fig. 3. Adjusting the weight in the output layer

Fig. 3 shows the learning process for one weight from neuron p in hidden layer j to neuron q in output layer k. The output of neuron of layer k, subtracted from the target value (Target), gives an error signal. It is multiplied by the derivative of the compressive function [OUT(1 - OUT)] calculated for this neuron of layer k, thus giving the value δ.

$$\delta = OUT(1 - OUT)(Target - OUT) \tag{2.2}$$

Then δ is multiplied by the OUT value of neuron j, from which the weight in question comes out. This product is in turn multiplied by the learning rate coefficient η (usually from

0.01 to 1.0), and the result is added to the weight. The same procedure is performed for each weight from the hidden layer neuron to the neuron in the output layer.

The following equations illustrate this calculation:

$$\Delta wpq,k = \eta\, \delta q,k\, OUT \tag{2.3}$$

$$wpq,k(n+1) = wpq,k(n) + \Delta wpq,k \tag{2.4}$$

where wpq,k(n) is the value of weight from neuron p in the hidden layer to neuron q in the output layer at step n (before correction); note that index k refers to the layer where this weight ends, i.e, according to the convention adopted in this book, with which it is combined; wpq,k(n+1) is the value of the weight at step n + 1 (after correction); δq,k is the value δ for neuron q, in the output layer k; OUTp,j is the value OUT for neuron p in the hidden layer j.

Adjusting the weights of the hidden layer. Consider one neuron in the hidden layer that precedes the output layer. As it passes forward, this neuron transmits its output to neurons in the output layer through the weights that connect them. During learning, these weights function in reverse order, passing the value δ from the output layer back to the hidden layer. Each of these weights is multiplied by the value of δ of the neuron to which it is connected in the output layer. The value δ required for a hidden layer neuron is obtained by summing all such products and multiplying by the derivative of the compressive function (see Fig. 4):

$$\delta_{q,k} = \mathrm{OUT}_{p,j}(1 - \mathrm{OUT}_{p,j})\left[\sum_{q} \delta_{q,k} w_{pq,k}\right] \tag{2.5}$$

When the value of δ is obtained, the weights feeding the first hidden level can be adjusted using equations (2.5) and (2.6), where the indices are modified according to the layer.

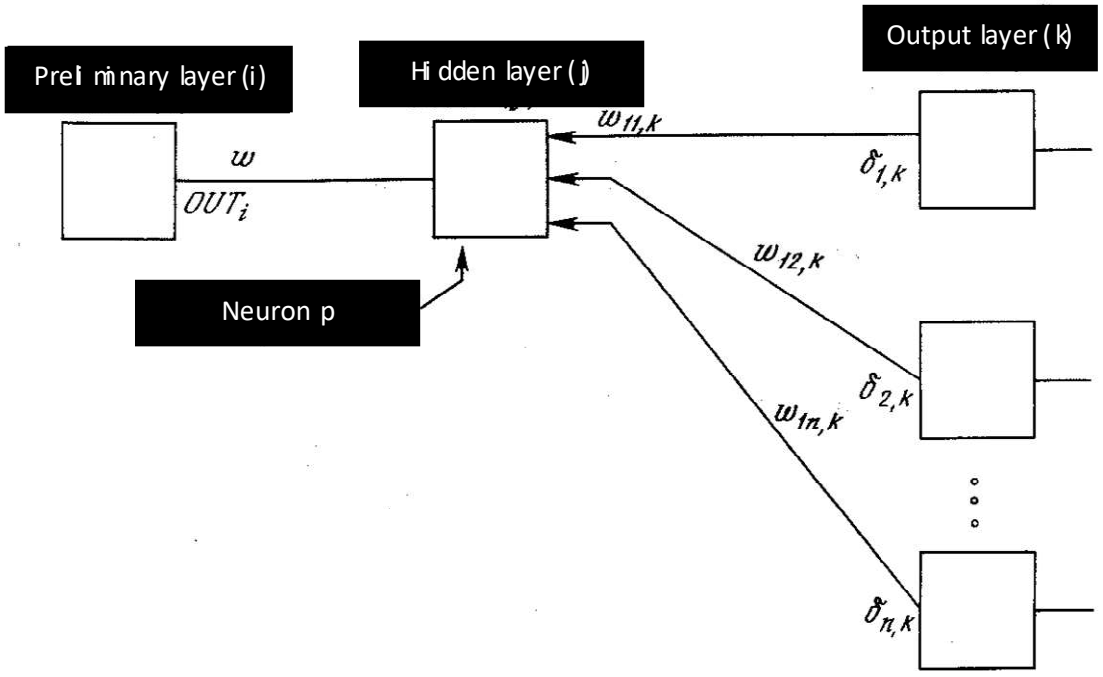

Fig. 4. Setting the weight in the hidden layer

For each neuron in a given hidden layer, δ must be calculated and all the weights associated with this layer adjusted. This process is repeated layer by layer towards the input, until all weights are adjusted.

Using vector notations, the error back propagation operation can be written in a much more compact way. Let us denote the set of values δ of the output layer by Dk and the set of

weights of the output layer as array Wk. To obtain Dj, the δ vector of the output layer, the following two operations are sufficient:

1.Multiply the o-vector of the output layer Dk by the transpose matrix of weights W'k connecting the hidden layer to the output layer.

2.Multiply each component of the resulting product by the derivative of the compressive function of the corresponding neuron in the hidden layer.

In the symbolic notation

$$Dj = DkW'k \$[0j \$(I – 0j)], \quad (2.6)$$

where the operator $ denotes the component product of vectors, Oj is the output vector of layer j, and I is a vector whose components are all equal to 1.

Adding a neural bias. In many cases it is desirable to endow each neuron with a learning bias. This allows to shift the origin of the logistic function, giving the effect similar to the adjustment of perseptron neuron threshold, and leads to the acceleration of the learning process. This feature can be easily introduced into the learning algorithm by adding a weight to each neuron, attached to +1. This weight is trained the same way as all other weights, except that the signal applied to it is always equal to +1, not the output of the previous layer neuron.

Pulse. The literature [24] describes a learning acceleration method for the backpropagation algorithm that also increases the stability of the process. This method, called pulse, consists in adding to the weight correction a term proportional to the magnitude of the previous weight change. Once a correction occurs, it is "remembered" and serves to modify all subsequent corrections. The correction equations are modified as follows:

$$\Delta wpq,k(n+1)= \eta\ \delta q,k\ OUTp,j + \alpha\Delta wpq,k(n) \quad (2.7)$$

$$wpq,k(n+1) = wpq,k(n) + \Delta wpq,k(n+1) \quad (2.8)$$

where (α – is the momentum coefficient, usually set around 0.9. Using the momentum method, the network tends to follow the bottom of narrow gullies of the error surface (if any), rather than moving from slope to slope. This method seems to work well on some problems, but has a weak or even negative effect on others.

There is a similar method based on exponential smoothing, which may have an advantage in a number of applications.

$$\Delta wpq,k(n+1)= (1-\alpha)\ \delta q,k\ OUTp,j + \alpha\Delta wpq,k(n) \quad (2.9)$$

Then the change in weight is calculated

$$wpq,k(n+1) = wpq,k(n) + \eta\Delta wpq,k(n+1), \quad (2.10)$$

where α the smoothing coefficient varies and ranges from 0.0 to 1.0. If equal to 1.0, the new correction is ignored and the previous correction is repeated. In the region between 0 and 1, the weight correction is smoothed by a value proportional to . Still, η is the learning rate coefficient, serving to control the average weight change.

For the development of the program was used (Type of implementing computer) PC (personal computer). The programming language was C++ Builder XE8. Operating system used: Windows XP, Vista, Windows 7, Windows 8, Windows 10.

For system training there was created a database (Certificate of Database Registration No. 2021622507 dated 09.11.2021) for storage and issuing of operative and reference information about condition of ionizing radiation sources (in places of radioactive waste disposal/storage) and toxic pollution at their quantitative content which is/are insignificantly exceeding operating hygienic norms/levels. The description of the database provides a list of information fields for a comparative analysis of toxic air pollution in the analyzed conditions of the residential area and residential area of comparison settlements. For each item, 32 measurements were made (presented in the database).

Indicators were analyzed in the computer program:
- Natural background radioactivity (values were analyzed);
- Character of soil (variants);
- Presence of forests (variants);
- Amount of precipitation;
- Radionuclides (excess of specific activity in the analyzed soil samples);
- Content of chemical pollutants in the air of the residential area (normal/exceeding the average daily MPC).

4. RESULTS

For the development and preparation of computer implementation of the proposed program, neural networks were chosen as a variant of implementation of artificial intelligence systems.

Any expert system, including the one under development, should conditionally consist of four blocks: an interface with the user, a knowledge base, a computing block, a block of explanations, allowing the user to trace the "course of reasoning" of the system in a particular case. The connecting element between these blocks is the method by which the expert system, in response to the user's request, produces a result (conclusion). The paper proposes a classification of such methods into three main groups:

1) methods of logical rules "in pure form", when the formalization of the rules of obtaining the result is carried out by the expert;

2) the same methods, but formalization of rules is carried out by the researcher, observing the work of the specialist from outside;

3) methods based on the principle "look and learn".

Creating even simple expert systems based on Methods 1 and 2 is a complex task, primarily because it requires the joint work of specialists of different profiles. Traditional expert systems based on knowledge bases and logical rules require quite a lot of time and money for creation. Creation of expert system can be divided into several stages.

1. Task setting: defining the objectives of the expert system, the set of input data and the form of presentation of the answer.
2. Data collection: a set of representative material for statistical research and its structuring - division into subgroups according to various attributes.
3. Statistical processing: identification of patterns linking the input data to the answer - calculation of averages and relative values, their comparison, correlation, regression, factor analysis, etc.
4. Creating a knowledge base: drawing up logical rules by which the expert system should work.
5. Algorithm programming: transferring logical rules to a programming language.
6. Creation of system interface: development of means of interaction of the system with the user - data input forms, response output, etc.
7. Debugging and testing: checking the program operation and testing in real conditions.

When creating logical expert systems, the 3rd, 4th and 5th stages take the most time and require the joint work of both subject specialists and programmers and mathematicians. Despite the emergence of computer tools for designing expert systems, the main work is entrusted to specialists. Several serious problems arise in this case.

The first one is that when solving complex real-world problems (medicine, biology), the number of logical rules increases significantly. Often there is such a complex system of interrelations between them that it simply cannot be comprehended. Breaking a problem into blocks does not always help either: firstly, it is not always easy to do, and secondly, when breaking it down, some connections may sometimes get lost.

The second and even more serious problem is that it is not always possible to express the computational process by logical rules. This may be due both to the complexity of the problem itself, and to the peculiarities of the subject specialist's activity. This is especially evident in medicine, where the decision-making process largely relies on the intuition and experience of a physician who is not an expert in his own thinking. In all of these cases, it is said that the task is not amenable to algorithmization. Moreover, even if the creators manage to develop an algorithm, there is never a sufficient guarantee that it will work correctly in real conditions, and this can be checked only after all the work on creating the system has been completed.

In the creation of the self-learning system developed by the authors (Certificate of Registration of Computer Software №2021681088 from 02.12.2021), you can identify several stages, some of which coincide with the stages of creating traditional systems.

1. Problem Statement. The same as for the traditional systems plus the choice of the optimal structure of a neural network and training methods (for the majority of problems the structure and methods are standard). \
2. Gathering of training data. A set of examples for training the network, each of which presents an array of input data and a corresponding response known in advance (Database Registration Certificate № 2021622507, 09.11.2021). Collection of data for further training of the network was carried out during 3 years for all selected parameters in conditions of different radiation background, chemical pollution, natural-climatic factors and their combinations, which entered into a single database.
3. Neural network creation and training. This stage does not require any statistical calculations. The training of the developed neural network was an automatic process, which only after its completion required the participation of a specialist to evaluate the results and compare them with the existing database. Schemes of problem statement, ways of data representation and ways of response production by the neural network are designed in such a way that most of the problems fit into these standard schemes.
4. Creation of interface of this program is the same as for traditional expert systems.
5. Debugging and testing. The stage included mainly debugging of program work, as testing was carried out in the process of network training.

The available advantages of this neural network expert system are shown when solving hard-to-algorithm problems:

1. A neural network makes decisions on the basis of experience, which it acquires independently. "Independently" in this case means that a user of expert system does not need to establish relationships between input data and necessary decision, spending time on various statistical processing, selection of mathematical apparatus, creation and checking of mathematical models.
2. The decision made by a neural network is not categorical. The network gives out the decision together with a degree of confidence in it, that leaves to the user an opportunity to critically estimate its answer.
3. A neural network allows to model a decision-making situation.
4. A neural network gives an answer very quickly (fractions of a second), which allows to use them in various dynamic systems, which require immediate decision-making.

It should be emphasized that application of implicit algorithms does not contradict and does not cancel use of formal methods, but can be complemented by them if necessary.

The program has additional functions: saving the results of the work; printing the results of the forecast; clearing the values of the indicators to be filled in.

The selection of indicators was carried out in various combinations taking into account the results obtained after training of artificial neural networks and evaluation of the results of their work (Fig. 5).

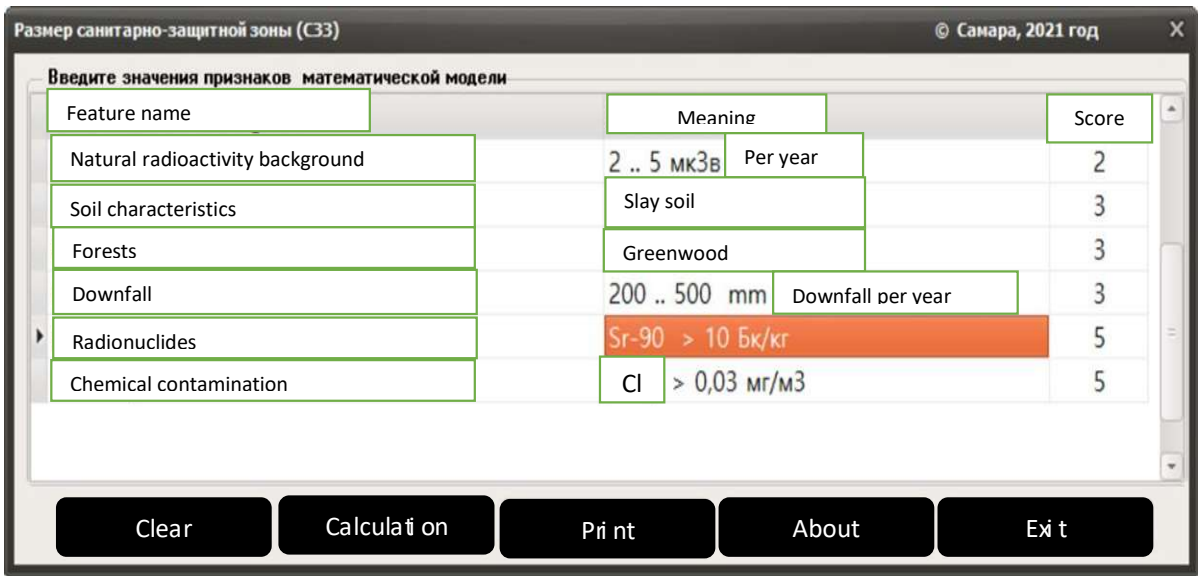

Fig. 5. Main screen of the computer program

Subsequent tests of the neural network showed that the quality of its work for an objective assessment was 80% of the correct solutions, which confirmed the possibility of using AI for mathematical modeling of the assessment of harm of personnel work in the conditions of sanitary protection zones, as well as calculating the minimum distances from the RAW object to the residential development in a particular area (Fig. 6).

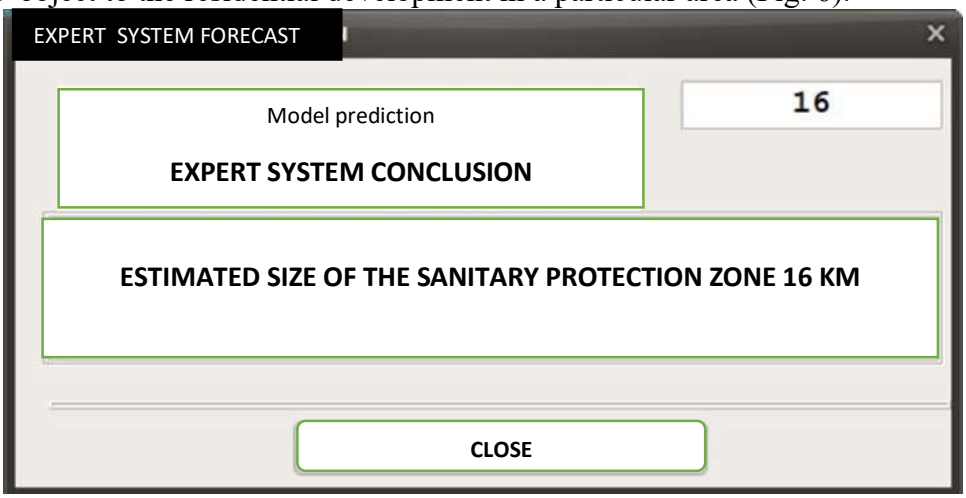

Fig. 6. Results screen

5. CONCLUSION

The developed computer program: "Automation of determination of dimensions of sanitary protection zones (SPZ) and surveillance zones for storage of sources of ionizing radiation in places of burial / storage of radioactive waste" was intended to automate the process of calculations of dimensions of sanitary protection zones (SPZ) and surveillance zones for storage of sources of ionizing radiation in places of burial / storage of radioactive waste based on mathematical regression models.

The main advantage of the developed neural network is that it can identify non-obvious attributes for determining the size of sanitary protection zones (SPZ). These algorithms do not rely on the available knowledge about the influence of the analyzed indicators, which makes them objective and allows to use them to create models of the state of storage of radiation sources.

REFERENCES

1. Artobolevsky, I.I., Vishnevsky, A.A. & Bykhovsky M.L. Information retrieval systems in medicine. *Machine diagnostics and information retrieval in medicine*.

2. Bartsev S.I. & Okhonin, V.A. (1986). Adaptive networks of information processing. *Preprint of Institute of Physics SB AS USSR*, **59**, 1–20.

3. Bedrekovsky, M.A., Gamkrelidze, S.A. & Fedchenko, O.I. (1991) Elemental base of neurocomputers. *Zarubezhnaya radioelektronika*, **6**, 45–49.

4. V.I. Chernov, I.E. Esaulenko, M.V. Frolov et. al. (2009) *Informatika. Kniga 1. Osnovi obschei informatiki* [Computer science. Book 2. Fundamentals of Medical Informatics]. Moscow, Russia: Drofa, [in Russian].

5. Duke W., Emanuel W. (2003). *Informacionnie tehnologii v mediko-biologicheskih issledovaniyah* [Information technologies in medical and biological research]. Saint Perersburg, Russia: Piter.

6. Federal Law On the Management of Radioactive Waste and Amendments to Certain Legislative Acts of the Russian Federation of 11.07.2011 N 190-FZ (last revision).

7. Gasparyan, S.A. (2001). Classification of medical information systems. *Information technologies in health care*, **10-12**.

8. Gassnikov, V.K. (1997). *Fundamentals of scientific management and informatization in health care: a training manual*. Izhevsk, Russia.

9. Gelfand, I.M., Huberman, Sh.A., Gindikin, S.G., et. al. (1985). Some problems of classification and prognostication from different fields of medicine. *Problems of Medical Diagnostics and Forecasting from the Viewpoint of Mathematics*, 110–125.

10. Gelman, V.Y. (2000). *Kompiuternie kommunikacii v medicine* [Computer communications in medicine]. Saint Petersburg, Russia, [in Russian].

11. Gelman, V.Y. (2001). *Medicinskaya informatika* [Medical informatics. Workbook.]. Saint-Petersburg, Russia: Piter, [in Russian].

12. Genkin, A.A. (1999) *Novie infomacionnie tehnologii analiza medicinskih dannih (kompleks program OMIS)* [New information technology of medical data analysis (OMIS program complex)]. Saint Peresrburg, Russia, [in Russian].

13. Gorban, A.N. *Obychenie neironnih setei* [Training of neural networks]. Moscow, Russia: Paragraph, [in Russian].

14. Gubler, E.V. (1978) *Chislennie metodi analiza i raspoznavaniya patologicheskih protsessov* [Computational methods of analysis and recognition of pathological processes]. Leningrad, USSR: Medicine, [in Russian].

15. Makarova, N.V. et. al. (2001) Informatics: textbook. Moscow, Russia.

16. *Information technologies of territorial management. Specialized issue "Telemedicine"*. Moscow, Russia: All-Russian Research Institute for Computer Technologies and Informatization. - T. 40. - 2003.

17. Kudrina, V. G. (1999) *Meditsinskaya informatika* [Medical informatics]. Moscow, Russia, [in Russian].

18. Marasanov, V.V. *Matematicheskie modeli differentsialnoi diagnostiki boleznei* [Mathematical models of differential diagnostics of diseases]. Kishinev, USSR: Shtintsa, [in Russian].

19. Masalovich, A.I. (1992) From neuron to neurocomputer. *Journal of Doctor Dobb*, **1**, 20–24.

20. Nemirko, A.P. (1991) *Avtomaticheskie sistemi dlya meditsinskogo i biologicheskogo issledovaniya* [Automated systems for medical and biological research]. Saint Petersburg, Russia, [in Russian].

21. Omelchenko, V.P. & Demidova, A.A. (2018) *Informatika. Praktikum* [Practice on medical informatics]. Moscow, Russa: GEOTAR-Media, [in Russian].

22. Postnova, T.B. (1972). *Informatsionnie sistemi i sistemi diagnostiki v meditsine* [Information and diagnostic systems in medicine]. Moscow, Russia: Nauka, [in Russian].

23. Rossiev, D.A. (1994). *Ekspertnie samoobuchayuschiesya sistemi neironnih setei v meditsine* [Neuro-networked self-learning expert systems in medicine]. Krasnoyarsk, Russia, [in Russian].

24. Rossiev, D.A. (1995). Self-learning neural network expert systems in medicine: theory, methodology, tools, implementation. *M.D. Thesis*, Biophysics. Krasnoyarsk.

25. Wasserman, P. (1989) *Neurocomputer Technique: Theory and Practice*. New York, NY: Van Nostrand Reinhold Co.

