MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
VN Karazin Kharkiv National University School of Mathematics and Computer Science
Department of Theoretical and Applied Informatics

Master's Thesis

Using a COVID-19 prediction neural network based on the open-source library KERAS

Final year Master's Program student,
group MCS-63
specialty - Computer Sciences and Information Technologies,
Author: Tian Zhong
educational program: "Informatics"
Supervisor: Ievgen Meniailov
Reviewer: Kseniia Bazilevych

Kharkiv, 2024

INTRODUCTION

The appearance of new pathogens and the rapid spread of new emerging diseases pose serious challenges to civilization, which require adequate methods and means of controlling epidemic processes. In such conditions, there is an urgent need for modeling and decision-making support tools based on adequate mathematical models aimed at predicting their consequences.

The purpose of the research isdevelopmentand analysismodels of epidemic processes based on machine learning methods for well-founded disease prevention and control strategies.

According to the goal, it is necessary to solve the following tasks:

– To conduct an analysis of machine learning methods for studying the epidemic process.

– Classify machine learning methods for solving problems of epidemiological diagnostics.

– Develop models of epidemic processes based on machine learning methods.

– Investigate the results obtained using the developed machine learning models.

Object of research:the epidemic process of the spread of the COVID-19 virus in Ukraine.

Subject of research:mathematical model and information technology for solving the problem of forecasting the spread of the COVID-19 virus in Ukraine.

1 CLASSIFICATION OF MACHINE LEARNING METHODS FOR MODELING EPIDEMIC PROCESSES

1.1 Machine learning

Machine learning is the use of mathematical data models that help a computer "learn" (that is, gradually improve its performance in a certain task) without direct instructions [1]. It is considered one of the forms of artificial intelligence. In machine learning with the help of algorithms, regularities in the data are revealed. Based on these patterns, a data model is created for forecasting. The more data such a model processes and the longer it is used, the more accurate the results become. The adaptive nature of machine learning makes it great for scenarios where data is constantly changing, query or task properties are unstable, or writing code to solve is virtually impossible.

Machine learning is used in many areas. Its capabilities are constantly expanding. Here are some of the main benefits that come from machine learning:

– Gaining insights – Machine learning helps identify patterns or patterns in both structured and unstructured data to provide valuable insights [2].

– Increasing the level of data integrity is an option for intelligent data analysis. That increases its accuracy and expands possibilities in dynamics [3].

– Empowering users – adaptive interfaces, targeted content, chatbots and virtual assistants with voice support are examples of how machine learning empowers users [4].

– Risk reduction – machine learning enables the tracking and detection of attackers' techniques so that action can be taken before any damage is done [5].

– With the help of machine learning, it is possible to perform intelligent analysis of data related to customers [6]. This allows you to identify patterns and

behaviors in order to optimize product recommendations and maximize customer service.

– Cost reduction – automation of processes, which frees up time and resources for the most important tasks [7].

Machine learning tasks generally fall into two broad categories, depending on whether the learning "signal" or "feedback" is available to the learning system:

1) Supervised learning [8]: Computers are presented with examples of inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. In some cases, the input signal may be available only partially, or be limited by special feedback:

– Semi-automatic learning – the computer is given only an incomplete training signal: a training set that lacks some (often numerous) target outputs [9].

– Active learning – the computer can only receive training labels for a limited set of instances (depending on the budget), and must also optimize its selection of objects to receive labels. For interactive use, they can be provided for marking to the user [10].

– Reinforcement learning – training data (in the form of rewards and punishments) is provided only as feedback on the actions of the program in a dynamic environment, such as when driving a car or playing a game with an opponent [11].

2) Learning without a teacher (spontaneous learning). The learning algorithm is not given labels, leaving it to find structure in its input. Learning without a teacher can be a goal in itself (revealing hidden regularities in data) or a means of achieving a goal (learning features) [12].

One of the most important tasks of data analysis is classification - assigning objects of the subject area to predetermined groups, called classes [13]. At the same time, each class should have objects that are similar in their properties. By generalizing the properties of known objects of the class to new objects assigned to

it, you can gain knowledge about them. The task of classification is solved with the help of analytical models called classifiers. Classifying an object means presenting a set of its features (usually represented as a vector) to the input of the classifier model, which must assign a label or class number to it. Currently, a large number of different types of classifiers have been developed, for the construction of which both statistical methods (logistic regression, discriminant analysis) and machine learning methods (neural networks, decision trees, etc.) are used. The need to use a large number of different classification methods in data analysis is due to the fact that the tasks solved with it may have their own characteristics, related, for example, to the presentation of the initial data, their quantity and quality, which requires the selection of an adequate classifier. Therefore, the choice of a classifier corresponding to the specifics of the analysis problem to be solved is an important factor in obtaining the correct solution.

Cluster analysis is the task of dividing a given sample of objects (situations) into subsets, called clusters, so that each cluster consists of similar objects, and the objects of different clusters differ significantly [14]. The clustering task belongs to statistical processing as well as to a broad class of unsupervised learning tasks. Cluster analysis is not a single algorithm, but a general problem for the solution of which various approaches are used. In particular, clustering algorithms can differ significantly in terms of understanding what to include in one cluster and how to effectively search for them. Among the popular concepts of clusters are groups with elements that are formed based on the distance between them, on the density of areas in the data space, intervals, or on specific statistical distributions. Therefore, clustering can be formulated as a multi-criteria optimization problem. The appropriate clustering algorithm and parameter selection (including parameters such as distance function, density threshold, or number of expected clusters) depend on the specific data set and intended use of the results. Cluster analysis as such is not an automated task, but an iterative process of knowledge discovery or interactive

multi-criteria optimization that involves trial and error. It is often necessary to change the data processing process and model parameters until a result with the specified properties is obtained.

Regression analysis is a search for a model of the dependence of one quantity on another, expressed in the regression function [15]. Regression analysis is used if the relationship between variables can be expressed quantitatively in the form of some combination of these variables. The resulting combination is used to predict the value that the target (dependent) variable can take, which is calculated on a given set of values of the input (independent) variables. In the simplest case, standard statistical methods such as linear regression are used for this. Unfortunately, most real-world models do not fit within the framework of linear regression. Thus, comprehensive methods are needed to predict future values. Regression analysis uses a chosen estimation method, a dependent variable, and one or more independent variables to create an equation that estimates the value of the dependent variable. A regression model includes input data that tells us how well the model estimates the dependent variable. Also used in regression analysis to analyze relationships and test assumptions. Regression analysis is used to solve the following types of problems:

- find out which independent variable is related to the dependent variable.
- understand the relationship between dependent and independent variables.
- predict the unknown values of the dependent variable.

1.2 Classification of tasks in machine learning and methods used in solving these tasks

The following classification (Figure 1.1) of machine learning tasks and methods can be given. Let's consider some of them.

Logistic regression or logit model (English logit model) is a statistical model that is used to predict the probability of the occurrence of an event by comparing it

with a logistic curve based on the values of many characteristics [16]. Returns the response as the probability of a binary event (1 or 0). This model is often used to solve classification problems. The resulting classification rules are linear classifiers.

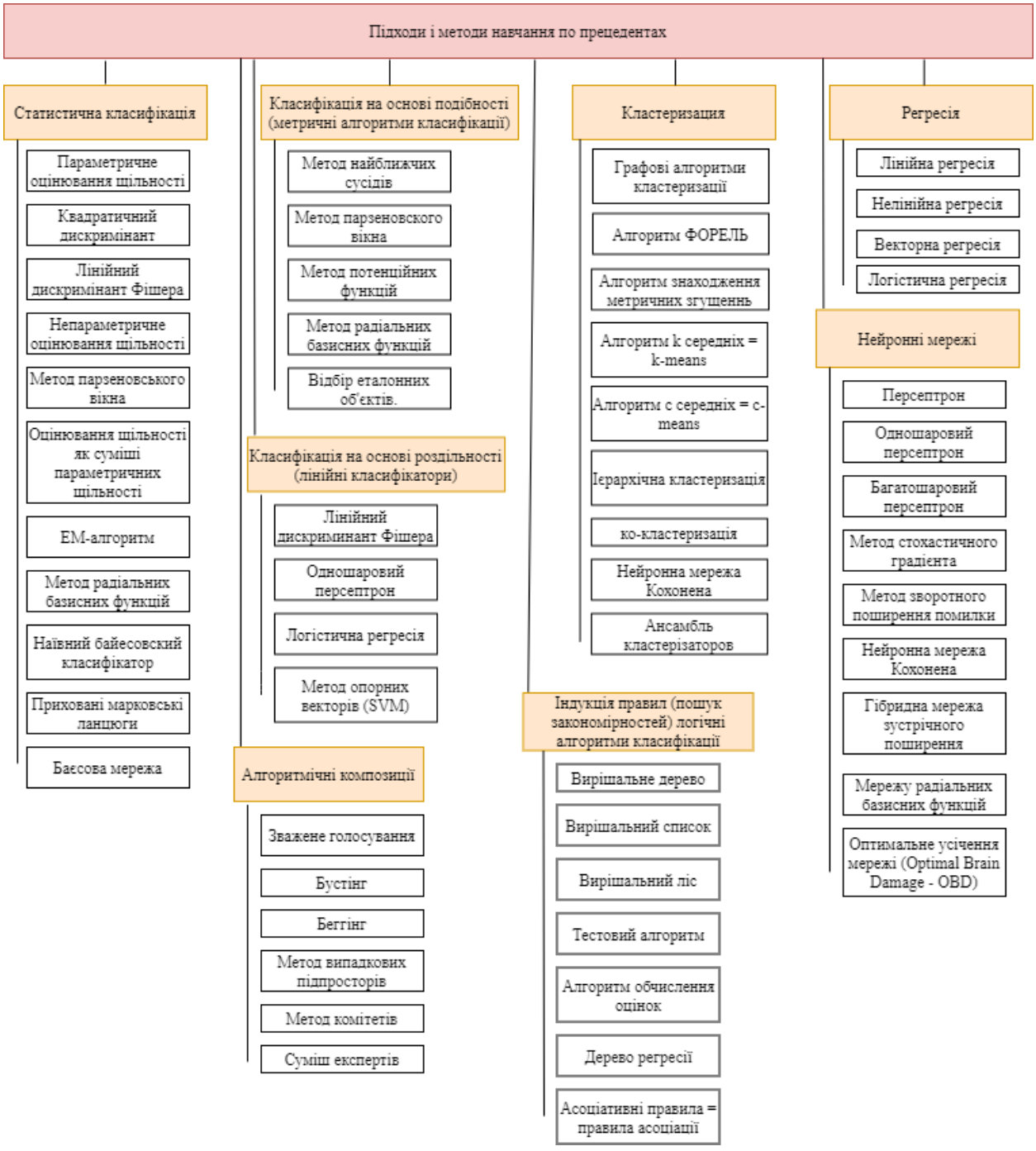

Figure 1.1 – Classification of learning methods by precedents.

The Bayesian approach is classic in pattern recognition theory and is the basis of many methods [17]. It is based on the theorem that if the class distribution densities are known, then the classification algorithm with the minimum error probability can be written explicitly. Due to its low quality of classification, the naive Bayesian classifier is mainly used either as a benchmark for experimental comparison of algorithms, or as an elementary building block in algorithmic compositions. Consider the frequent application of the Bayesian classifier to the task of classifying documents according to their content, namely to the classification of e-mails into two classes - spam (S) and non-spam (¬S), assuming that the probability of words in the text does not depend on each other.

A Bayesian network, a Bayes network, a belief network, a Bayesian model, or a probabilistic directed acyclic graph model is a probabilistic graph model (a type of statistical model) that represents a set of random variables and their conditional dependencies using a directed acyclic graph [18]. For example, a Bayesian network can represent probabilistic relationships between diseases and symptoms. Such a network can be used to calculate the probabilities of the presence of various diseases given the existing symptoms. Formally, Bayesian networks are a directed acyclic graph whose vertices represent random variables in the Bayesian sense: they can be observables, latent variables, unknown parameters, or hypotheses. Edges represent conditional dependencies; unconnected vertices (such that there is no path from one variable to another in the Bayesian network) represent variables that are conditionally independent of each other. Each vertex is associated with a probability function that takes as input a set of parent vertex values and outputs (as an output) the probability (or probability distribution, if applicable) of the variable represented by that vertex. Similar ideas can be applied to undirected and possibly cyclic graphs such as Markov networks. There are efficient algorithms that perform inference and training in Bayesian networks. Bayesian networks that model sequences of variables (for example, speech signals or protein sequences) are called dynamic Bayesian

networks. Generalizations of Bayesian networks that can represent and solve decision-making problems under conditions of uncertainty are called influence diagrams

The Parzen window method is a Bayesian classification method based on non-parametric reconstruction of the density based on the available sample [19]. After introducing the metric, the Parzen window method can be used without relying on the probabilistic nature of the data. The approach is based on the idea that the density is higher at those points near which there is a large number of sample objects. If the power of a set of elementary finals is much smaller than the sample size, then we can also take the histogram of the sample values as the density recovered from the sample. In another case (for example, continuous), this approach is not applicable, since the density is concentrated near the training objects, and the distribution function undergoes sharp jumps. It is necessary to use the Parzen-Rosenblatt recovery method.

Radial basis functions (RBF) are a set of hard interpolation methods; this means that the surface must pass through each measured reference value [20]. There are five different basis functions: plane spline, tension spline, fully regularized spline, multiquadric function, inverse multiquadric function. Each basis function has a different shape and allows obtaining different interpolated surfaces. RBF methods are a special case of splines. Conceptually, radial basis functions are reminiscent of placing a rubber membrane on measured reference points and simultaneously reducing the overall curvature of the surface. The selection of the basis function determines how the rubber membrane will be located between the values. The following diagram conceptually shows the plotting of a radial basis function surface on a series of elevation reference points. Note that the cross-sectional surface passes through the data values.

The method of k-nearest neighbors is used to solve the classification problem [21]. It assigns objects to the class to which most of its k nearest neighbors in the

multidimensional feature space belong. This is one of the simplest algorithms for learning classification models. The number k is the number of neighboring objects in the space of features that are compared with those classified by the object. In other words, if k = 10, then each object is compared with 10 neighbors. The method is widely used in Data Mining technologies. In the process of learning, the algorithm simply remembers all feature vectors and their corresponding class labels. When working with real data, i.e. observations whose class labels are unknown, the distance between the vector of the new observation and earlier after successful registration is calculated. Then k vectors closest to it are selected, and the new object belongs to the class to which most of them belong. The choice of parameter k is controversial. On the one hand, increasing its value increases the reliability of classification, but at the same time, the boundaries between classes become less clear. In practice, good results are given by heuristic methods for choosing the parameter k, for example, cross-validation. Despite its relative algorithmic simplicity, the method shows good results.

Unlike most existing clustering algorithms, the C-Means algorithm is fuzzy - each object is not uniquely included in any cluster, but belongs to all clusters with different degrees of membership [22]. This gives advantages in the quality of partitioning in cases where the clusters are close to each other and a large number of points are on their borders. However, the price of such vagueness is higher computational cost than that of such clear algorithms as K-Means. If such a shortcoming as the a priori determination of the number of clusters is preserved, there is a possibility that there is no guarantee of the global optimum of the result.

The decision tree is a hierarchical tree-like structure consisting of a rule of the form "If ..., then ..." [23]. Due to the training set, the rules are generated automatically during the training process. Unlike neural networks, trees as analytical models are simpler because the rules are generated in natural language: for example, "If the ad brought 1000 customers, then it is configured well." The rules are

generated by summarizing many separate observations (study examples) describing the subject area. Therefore, they are called inductive rules, and the learning process itself is called the induction of decision trees. Decision trees are used to support management decision-making processes used in statistics, data analysis, and machine learning. The tool helps to solve the following tasks:

– Classification. Assignment of objects to one of the previously known classes. The target variable must have discrete tasks.

– Regression (numerical prediction). Prediction of the numerical value of the independent variable for a given input vector.

– Description of objects. A set of rules in the decision tree allows you to compactly describe objects. Therefore, instead of complex structures used to describe objects, decision trees can be stored.

CART (Classification and Regression Tree) is an algorithm for learning decision trees that allows using both discrete and continuous target variables, i.e. solving both classification and regression tasks [24]. The essence of this algorithm is the usual construction of a decision tree.

The AdaBoost algorithm (abbreviated from adaptive boosting) is a meta-algorithm that, in the process of learning, builds a composition of basic learning algorithms to improve their efficiency [25]. AdaBoost is an adaptive boosting algorithm in the sense that each subsequent classifier is built on objects that are poorly classified by previous classifiers. The algorithm can be used in combination with several classification algorithms to improve their performance. AdaBoost boosts classifiers by combining them into a "committee". The algorithm is adaptive in the sense that each subsequent committee of classifiers is built on objects incorrectly classified by previous committees. AdaBoost is sensitive to data noise and outliers. However, it is less prone to overtraining compared to other machine learning algorithms.

TNT (A Tree-Based Pipeline Optimization Tool) is an open source project that was launched only a month ago (which is in its infancy for such systems) and is now actively developing. On the project website, the author calls on the community [26]. The purpose of TNT is to automate the construction of machine learning pipelines by combining a flexible representation of pipelines in the form ofexpression treeswith stochastic search algorithms such asgenetic programming.

The EM-algorithm (English Expectation-maximization) is an algorithm used in mathematical statistics to find estimates of the maximum likelihood of the parameters of probabilistic models, in the case when the model depends on some hidden variables [27]. As a rule, the EM algorithm is used to solve two types of problems. The first type includes tasks related to the analysis of truly incomplete data, when some statistical data are missing for any reason. The second type of problems includes those problems in which the likelihood function has a form that does not allow for convenient analytical research methods, but allows serious simplifications if additional "unobserved" (hidden, latent) variables are introduced into the problem. Examples of applied problems of the second type are the problems of pattern recognition and image reconstruction. The tasks are the mathematical essence of these taskscluster analysis,classificationandseparation of mixtures of probability distributions.

The method of support vectors is a technique of machine learning with a teacher [28]. It is used in classification and can be applied to regression tasks. One of the most popular learning methods used to solve classification and regression problems. The main idea of the method is to build a hyperplane that divides the objects of the sample in an optimal way. The algorithm works under the assumption that the greater the distance (gap) between the separating hyperplane and the objects of the divided classes, the smaller will be the average error of the classifier.

Apriori is one of the most popular algorithms for finding associative rules [29]. Due to the use of the anti-monotonicity property, it is able to process large

amounts of data in an acceptable time. Modern databases have very large sizes, reaching gigabytes and terabytes, and a tendency to further increase. And therefore, for findingassociative ruleseffective ones are neededscalable algorithms, which allow you to solve the problem in an acceptable time.

PageRank ("Page rank"; fromEnglishPage rank - rankingPaige) is one ofalgorithmswithstrong ranking. The algorithm is applied to a collection of related documentshyperlinks(such asweb pageswiththe world wide web), and assigns each of them some numerical value that measures its "importance" or "authority" among other documents. The algorithm can be applied not only to web pages, but also to any set of objects connected to each other by mutual links, that is, to anycount.

Also, in statistics, machine learning and information theory, there is a process of reducing the number of random variables by obtaining a set of main variables - this is dimensionality reduction. This process can be divided into feature selection (the process of finding a subset of original variables, features, or properties) for use in model building and feature extraction.

Conclusions to section 1

In this section, an analysis was carried out and a classification of modern mathematical models, computational methods and their applied information technologies, which are used in statistical data processing and medical data analysis, was implemented. The listed methods have their own characteristics, which makes it possible to apply them to various tasks, in particular, medical ones. To improve their quality and reduce computing costs, as well as memory costs, you can use methods of selecting the most informative features and filling in missing values from the point of view of the given task.

2 APPLICATION OF NEURAL NETWORKS TO THE MODELING OF EPIDEMIC PROCESSES

2.1 Architecture of neural elements

Neural network information processing technologies make it possible to create adaptive systems in which data processing is carried out using parallel association operations. Association rules are generated by the system itself, learning from examples and adjusting its functioning based on the results of its activities. The main element of an artificial neural network is a neural element or a formal neuron that performs the operation of non-linear transformation of the sum of products of input signals into weighting coefficients.

The neural element used in the modeling of artificial neural networks can be represented by the diagram shown in Figure 2.1.

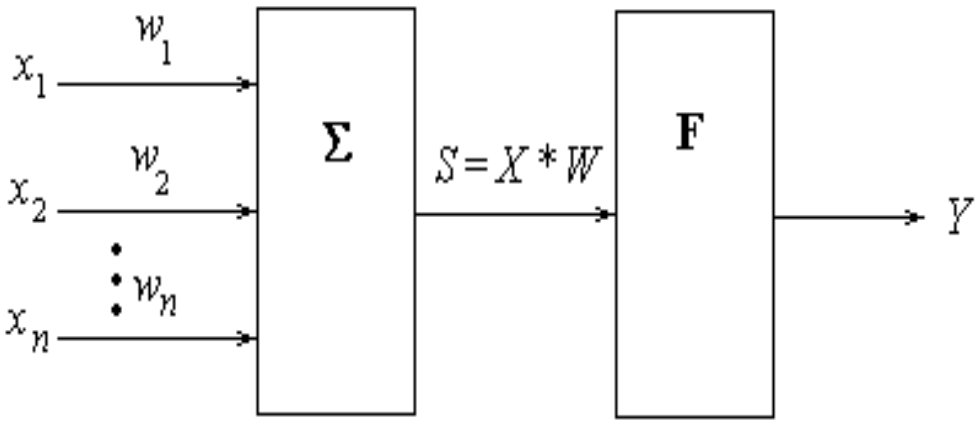

Figure 2.1 – Neuronal element

Each component of the input signal vector$X$=(x1,x2,...,xn), representing the output signals of other neural elements or the input of the network, is multiplied by the corresponding weight of the connection of the weight vector

W=(w1,w2,...,wn)and enters the block adder. The link weight is a scalar value, positive for reinforcing and negative value for inhibiting links. The level of excitation of the neural element is equal to $S = \sum_{i=1}^{n} w_i \cdot x_i$ or in vector form S = X * W. The weighted sum S is the scalar product of the vector of weights by the input vector:

$$S = \sum_{i=1}^{n} w_i \cdot x_i \cdot = |w| \cdot |x| \cdot \cos \alpha \,, \tag{2.1}$$

where | w |, | x | are the lengths of vectors W and X, respectively, and α is the angle between these vectors.

The lengths of the vectors are determined through their coordinates:

$$|W| = \sqrt{w_1^2 + w_2^2 + ... + w_n^2} \,, \tag{2.2}$$

$$|X| = \sqrt{x_1^2 + x_2^2 + ... + x_n^2} \,.$$

If the input vectors are normalized and | X | = const, then the value of the weighted sum will depend only on the angle between the vectors X and W. With different input signals, the weighted sum will change according to the cosine law. It reaches its maximum value when the input and weight vectors are collinear. The output signal Y is determined by transforming the sum S by the nonlinear activation function F:

$$Y=F(\sum_{i=1}^{n} wi \cdot xi). \tag{2.3}$$

Absolute values of weights wi, i = 1,2, ..., n, as a rule, belong to the interval [0; 1], which avoids large input values for other neural elements.

A neural network that receives a certain signal at the input, is able to produce a result after passing it through the neurons, depends on the weighting coefficients of all neurons. If S0 is the displacement (threshold) of the neural element characterizing the displacement of the activation function along the abscissa axis, then the weighted sum

$$S = \sum_{i=1}^{n} w_i \cdot x_i \text{ -S0.} \qquad (2.4)$$

Learning a neural network is nothing more than "tuning" the weights so that a certain input signal corresponds to a certain output signal. A training sample for neural networks is a set consisting of strings with training examples.

2.2 Activation functions of neural elements

The most common activation functions, nonlinear gain characteristics of a neural element, or transfer functions are: threshold, signum, logistic, hyperbolic tangent, linear, linear bounded, radial basis, etc.

The threshold binary function is a function for which the neural element remains inactive until the input reaches the threshold value S0 (Fig. 2.2):

$$Y(s) = \begin{cases} 0, S \le S_0 \\ 1, S > S_0 \end{cases} \qquad (2.5)$$

If S0 = 0, then the binary threshold function is called a unit activation function with a hard limit (hardlim (S)).

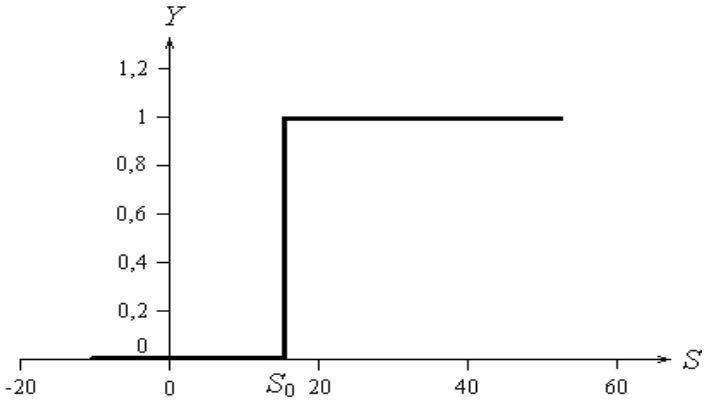

Figure 2.2 – Threshold binary function

The signum, or modified threshold function, for which the value of S0 = 0 (Fig. 2.3), is given by the equation:

$$Y(s) = \begin{cases} -1, S < S_0 \\ 0, S = 0 \\ 1, S > S_0 \end{cases}. \qquad (2.6)$$

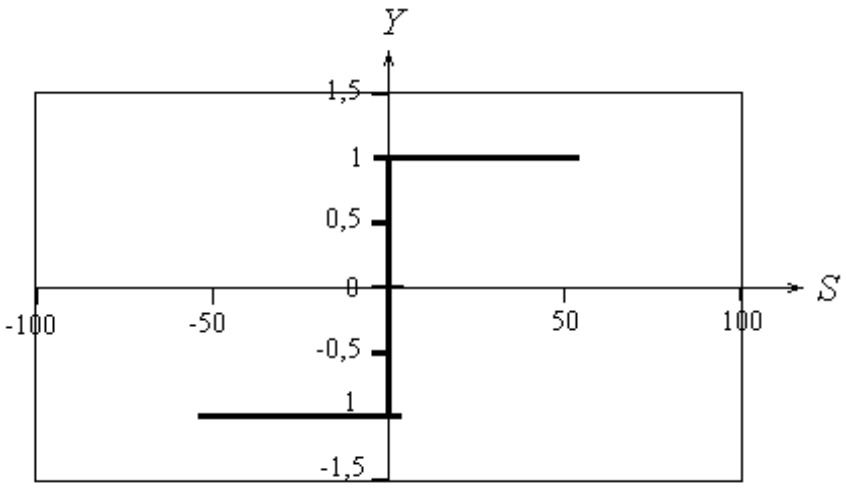

Figure 2.3 – Signum function

The sigmoid logistic function (S-shaped, having two horizontal asymptotes and one inflection point) is an increasing compressive function, the values of which belong to the interval (0; 1) (Fig. 2.4):

$$Y(s) = \frac{1}{1 + \exp(-c \cdot s)}, \tag{2.7}$$

where z> 0 is the coefficient characterizing the steepness of the logistic function that amplifies weak signals (logsig (S)).

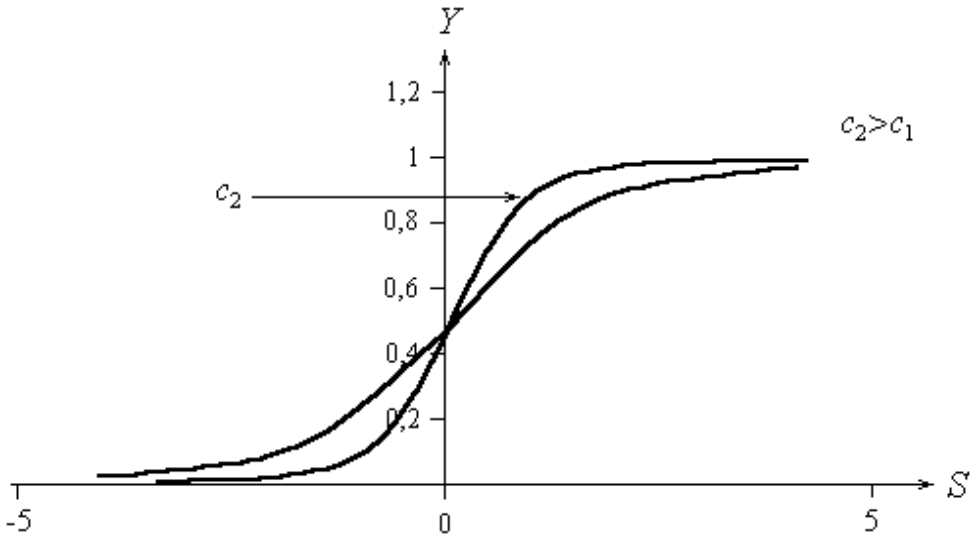

Figure 2.4 – Logistic function

In the general case, the logistic function is defined by the equation:

$$Y(s) = \frac{1}{1 + \exp(-c \cdot (s - s_0))}, \tag{2.8}$$

where S0 is the displacement value.

If the value of the coefficient c is large, then the graph of the general logistic function approaches the threshold function (Fig. 2.5).

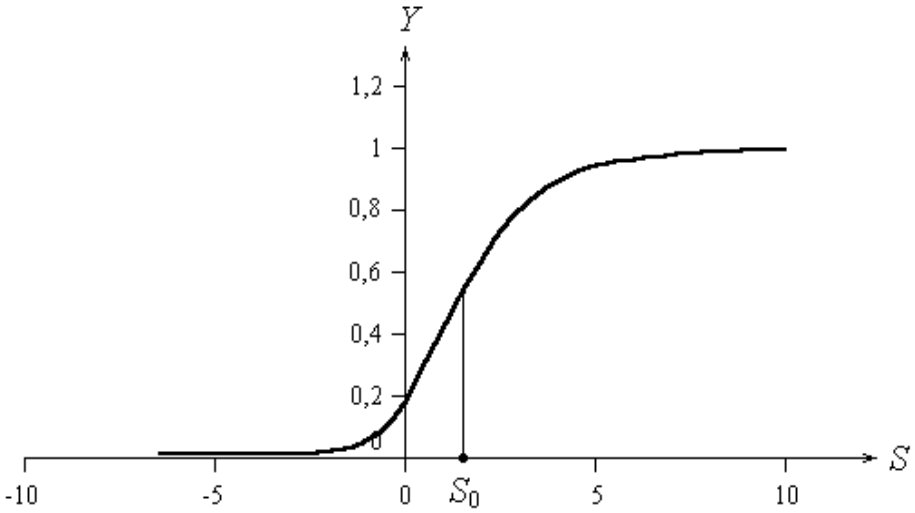

Figure 2.5 – General logistic function

Bipolar sigmoid function (Fig. 2.6), whose equation is:

$$Y(s) = \frac{2}{1 + \exp(-c \cdot s)} - 1, \tag{2.9}$$

takes values in the range (-1; 1):

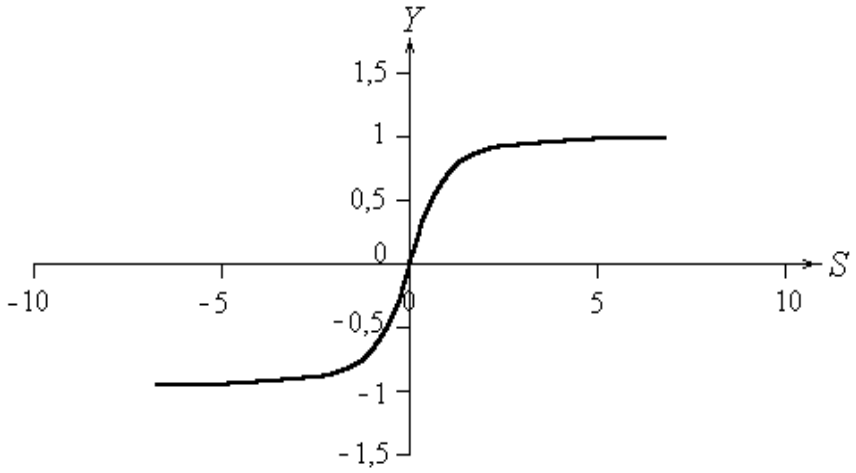

Figure 2.6 – Bipolar logistic function

The hyperbolic tangent (Fig. 2.7) is similar to the bipolar logistic function without displacement and is a symmetric function (tansig (S)):

$$Y(s) = \frac{\exp(c \cdot s) - \exp(-c \cdot s)}{\exp(c \cdot s) + \exp(-c \cdot s)} \qquad (2.10)$$

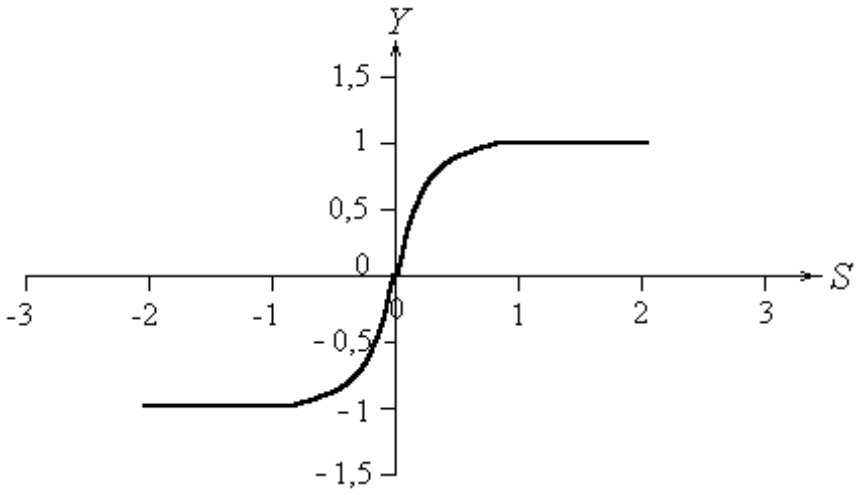

Figure 2.7 - Hyperbolic tangent

The linear activation function, the equation of which is Y (s) = k * s, where k is the slope angle of the straight line, is presented in Figure 2.8 (purellin (S)).

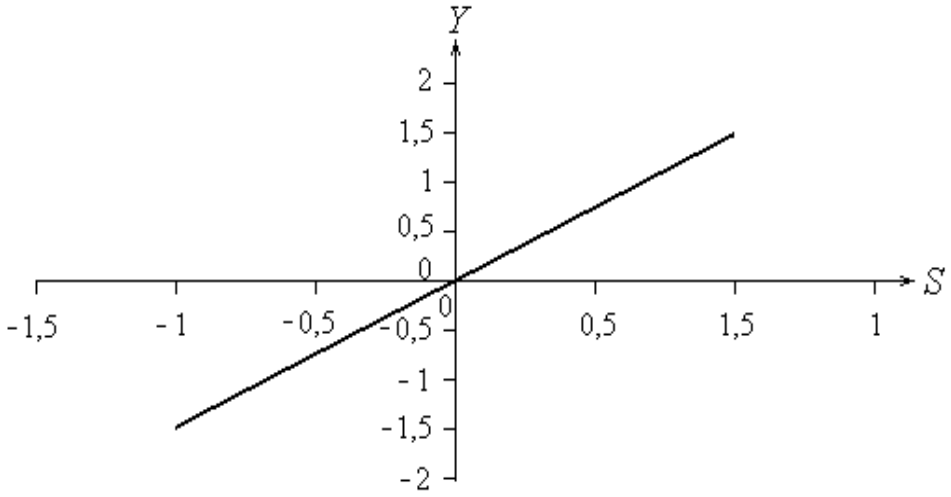

Figure 2.8 – Linear function

Linear, limited to the segment [-a; a] the activation function (Fig. 2.9) is determined by the formula:

$$Y(s) = \begin{cases} p, s > a \\ -p, s < -a \\ s, -a \le s \le a \end{cases} \quad (2.11)$$

where p is usually equal to one (satlin (S)):

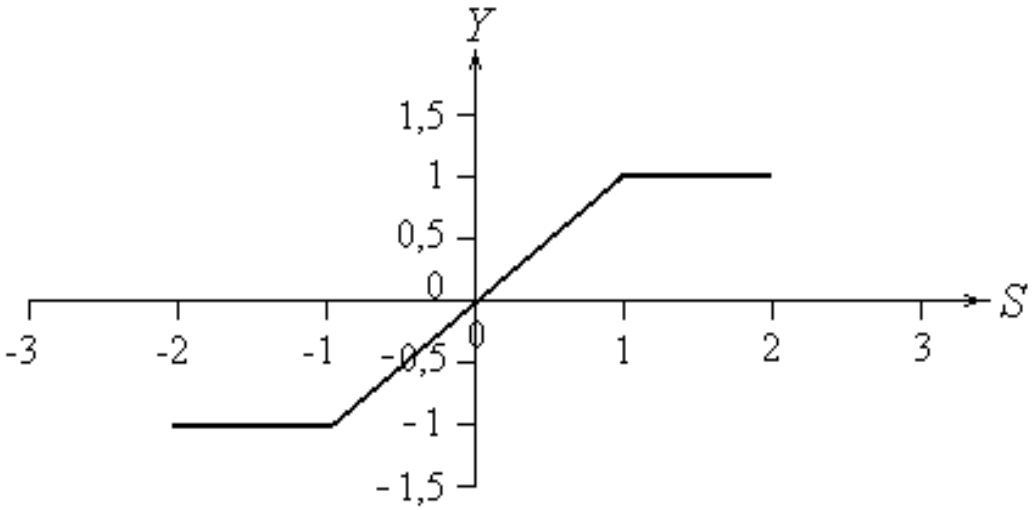

Figure 2.9 – Linear bounded function

The radial basis activation function (radbas (S)) is characterized by a Gaussian function for the normal distribution law, according to which:

$$Y(s) = \exp\left(\frac{-s^2}{2 \cdot \sigma^2}\right), \quad (2.12)$$

where σ- root mean square deviation characterizing the steepness of the radial basis function (Fig. 2.10). The value s is determined according to the Euclidean distance between the input and weight vectors:

$$S^2 = |XW|2 = \left(\sum_{i=1}^{n} (x_i - w_i)^2\right). \quad (2.13)$$

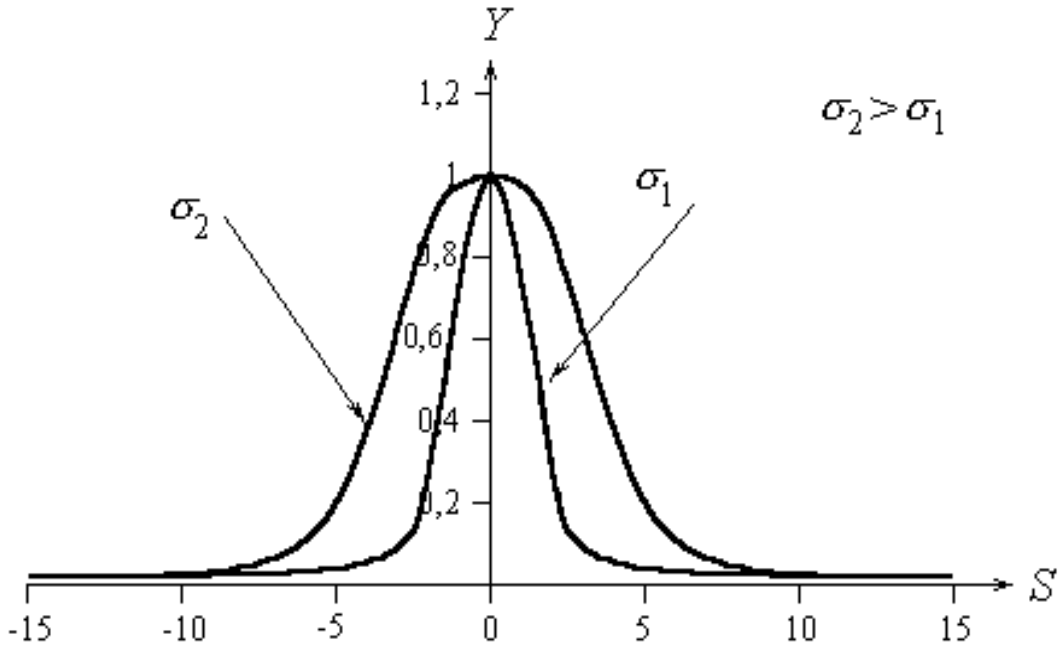

Figure 2.10 – Radial-basis function

Along with the listed functions, other activation functions can be used, for example, the logarithmic function.

Multivariate radial distributions allow a multivariate analysis to be performed by reducing it to the analysis of one-dimensional symmetric distributions, such as the multivariate normal distribution or uniform on a sphere centered at the origin.

2.3 Single-layer neural networks

A neural network is a collection of formal neurons connected in a certain way to each other and to the external environment. The vector of the input signal, which encodes the input effect or the image of the external environment, is fed to the network when the input neural elements are activated. The weights of connections of neural elements are represented in the form of a matrix W, for which wij is the weight of the connection of the jth neuron with the i-th. In the process of functioning of the network, the input vector is transformed into the output vector. The type of transformation is determined not only by the characteristics of neural elements, but

also by the features of the organization and the rules of learning the network (topology of connections), the presence or absence of competition, the direction of information flows, and others. Learning rules determine changes in connections and weights in response to input.

A layer of a neural network is a set of neural elements to which information from other neural elements of the network arrives in parallel at each time cycle. Single-layer networks include one layer of elements that processes input information (Fig. 2.11).

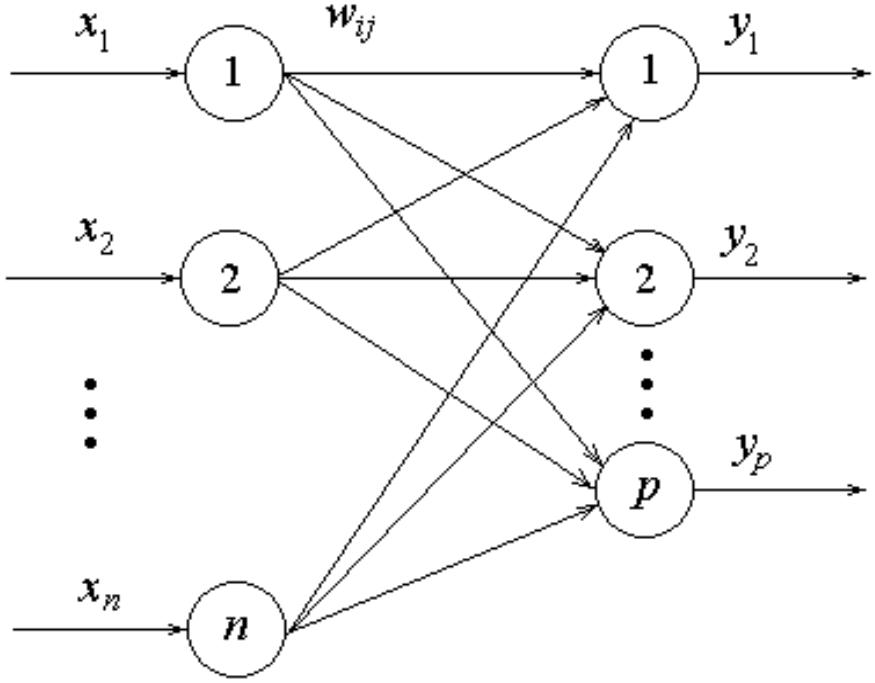

Figure 2.11 – Topology of a single-layer network

Each neuron of the distribution layer has synaptic connections with all neurons of the processing layer. The output value of the jth neuron of the processing layer of the network can be imagined as:

$$Y_j = F(s_j) = F\left(\sum_{i=1}^{n} w_{ij} x_i - S_{0j}\right), \qquad (2.14)$$

where is the displacement of the jth neural element of the output layer,S_{0j}

wij is the strength of the connection between the j-th neuron of the distribution layer and the i-th neuron of the processing layer.

The column vector of the weighted sum in matrix form is defined as follows:

$$S = W^{T}X\text{-}S_0, \tag{2.15}$$

where W is an n*p dimension matrix,

is a column vector of displacements of elements of the processing layer.S_0

A single-layer perceptron is a network consisting of S, A and R neural elements (Fig. 2.12), proposed by F. Rosenblatt in 1959, refers to single-layer networks.

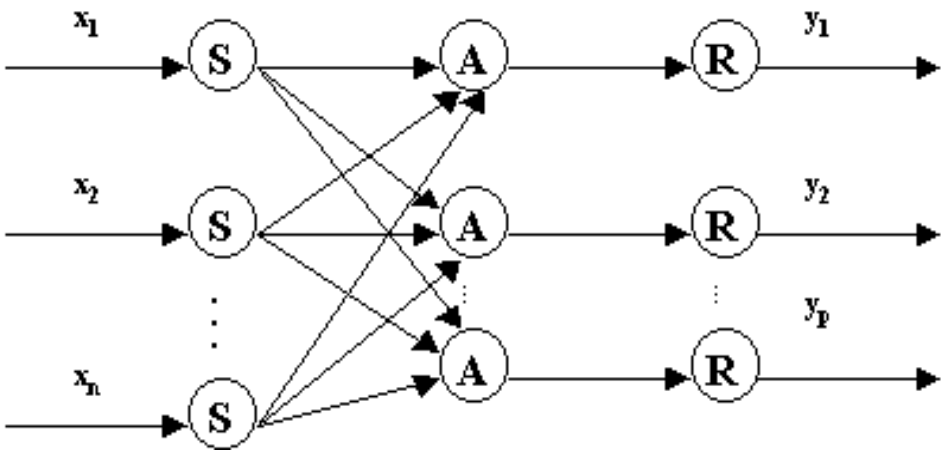

Figure 2.12 –The structure of the perceptron

Layer S neurons are called sensory and are designed to generate input signals as a result of external influences. Layer A neurons are designed for direct processing of input information and are called associative. Neurons of the R layer are called effectors and are introduced to transmit excitation signals to the corresponding object. A binary threshold function is used as the activation function of the

associative layer A. The training of the perceptron is carried out by adjusting the weight coefficients W between layers S and A. The elements of layer S perform purely distributional functions. Rosenblatt's learning rule, which is a modification of Hebb's rule, can be represented in the following form:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \cdot x_i \cdot t_j, \tag{4.16}$$

where tj is the reference or target value of the jth output of the neural network,

α– network learning rate, 0 <α<= 1.

Training is carried out with reinforcement, i.e. the weight coefficients of the network change only if the initial value does not coincide with the target value tj. The idea of a perceptron was used to recognize symbolic information in the simplest neurocomputer. The number of decisive elements of layer A of the perceptron coincides with the number of recognized classes.

2.4 Classification of neural networks for the analysis of epidemic processes

Depending on different characteristics, neural networks can be classified in different ways:

1. According to the type of input information, the following are distinguished:

a) analog neural networks using real numbers as output data;

b) discrete networks that operate with data presented in the binary number system.

2. Networks are distinguished by the nature of training:

a) with the teacher, when the original solution space of the neural network is known;

b) without a teacher, when the network forms the original solution space based only on input vectors. Such networks are called self-organizing networks.

3. According to the nature of connection settings, the following are distinguished:

a) networks with fixed connections, when the weighting coefficients of the network are selected immediately from the condition of the problem;

b) networks with dynamic connections, for which the setting of connections is carried out during the training process.

4. In the direction of information flows, the following are distinguished:

a) feed-forward networks, in which information is spread from layer to layer (Fig. 2.13):

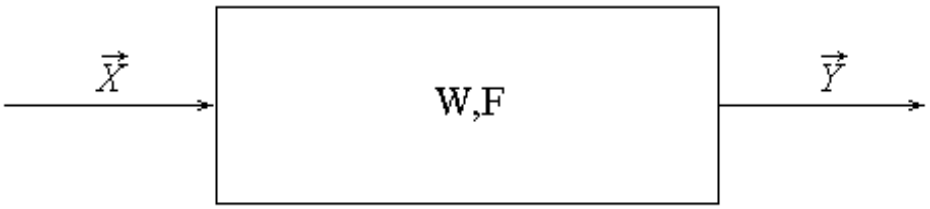

Figure 2.13 – Diagram of a direct signal transmission network,
W is the weight matrix, F is the nonlinear network transformation operator

b) feed-back networks characterized by both direct and reverse data transmission between network layers. Such networks include relaxation and multilayer networks in which there is no relaxation process.

Relaxation networks are networks in which data processing is carried out until the output values of the network stop changing (at a given accuracy), this state is called the equilibrium state. Such networks include Hopfield networks, bidirectional associative memory. The Hopfield network is characterized by a single feedback (Fig. 2.14).

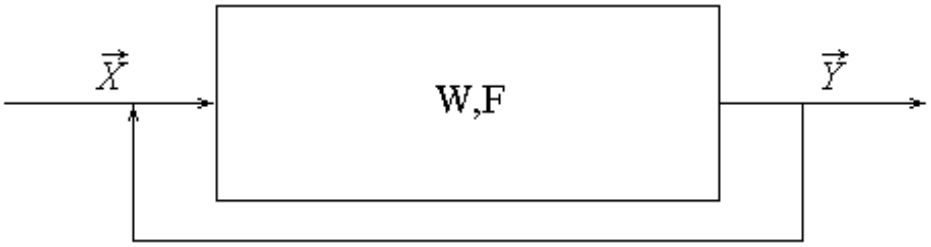

Figure 2.14 – Hopfield network scheme

Bidirectional associative memory is represented by a structure with non-unitary feedback (Fig. 2.15).

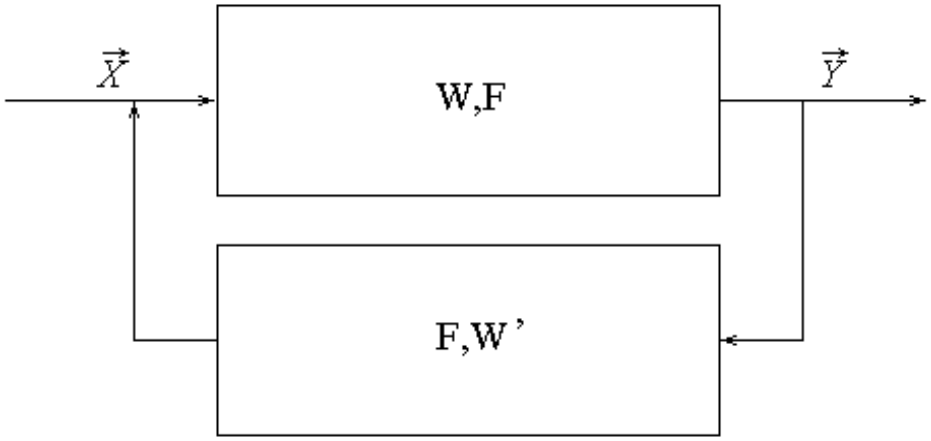

Figure 2.15 – Diagram of bilateral associative memory

Non-relaxation multilayer networks are based on multilayer perceptrons, and the basis of their training is the method of error back propagation. They are divided into recurrent (Fig. 2.16) and recirculation networks (Fig. 2.17).

Recirculation networks include Elm networks used for processing time series.

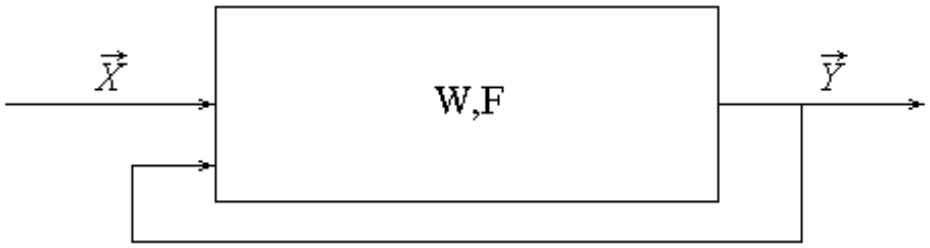

Figure 2.16 – Scheme of a recurrent network

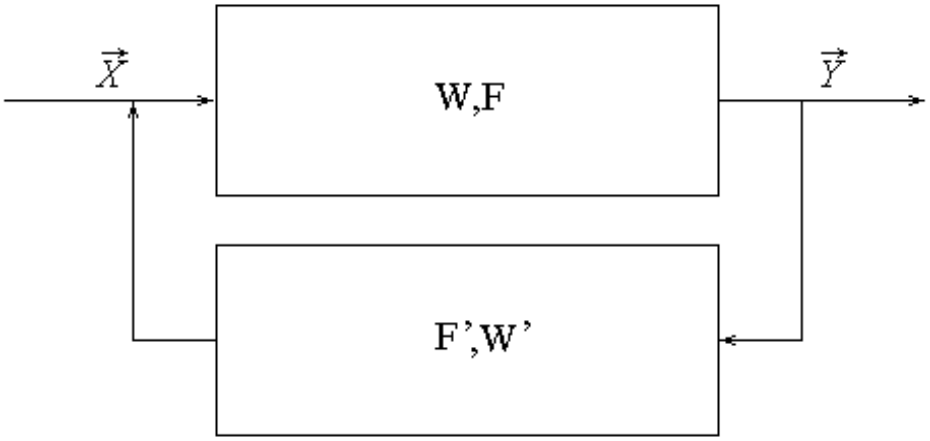

Figure 2.17 – Recirculation neural network scheme

In recurrent neural networks, the output values are determined depending on both the input and previous output values of the network. Recirculating neural networks are characterized by both forward and reverse transformation of information, and learning is carried out without a teacher (networks that self-organize during operation). Counterpropagation networks also include self-organizing networks in the course of their work.

5. The following are distinguished by teaching methods:

a) networks that learn by the method of backpropagation of the error;

b) networks with competitive training;

c) networks that self-organize in the process of work.

2.5 Learning neural networks with different architectures

Adaptation and self-organization of artificial neural networks is achieved in the process of their training. The goal of learning neural networks is to adjust their weights in such a way that would provide the required set of outputs for a certain set of inputs.

When solving applied problems with the help of neural networks, it is necessary to collect a representative amount of data in order to teach the neural network how to solve such problems. A training data set is a set of observations containing characteristics of the object under study. The initial selection of features is carried out on the basis of available experience, taking into account the opinion of experts. The question of how many observations are necessary to train a network is often a tricky one. A number of heuristic rules are known that establish a relationship between the number of observations used for training and the size of the network. The simplest of them says that the number of observations should be 10 times the number of connections in the network. The learning process is the process of determining the parameters of a model of a process or phenomenon implemented by a neural network. The training error for a particular network configuration is determined after running through the network all available observations and comparing the output values to the target values in the case of teacher training. The corresponding differences make it possible to form the so-called error function. If the error of the network, the output layer of which has n neurons, is the difference between the real and desired signals at the output of the i-th neuron, then the following functions can be used as error functions:$e_i = y_i - t_i$

– sum of squared errors sse $= \sum_{i-1}^{n} e_i^2$,

– mean squared error mse $=\frac{1}{n}\sum_{i-1}^{n} e_i^2$,

– adjustable or combination error msereg$=\frac{\gamma}{n}\sum_{i=1}^{n} e_i^2+\frac{1-\gamma}{n}\sum_{i=1}^{n} e_i^2$, where is the adjustment parameter,γ

– mean absolute error mae $=\frac{1}{n}\sum_{i=1}^{n}|e_i|$.

When modeling neural networks with linear neuron activation functions, it is possible to build an algorithm that guarantees the achievement of an absolute minimum of learning error. For neural networks with non-linear activation functions, it is generally not possible to guarantee that the global minimum of the error function will be reached. The surface of the error function is defined as a collection of points-values of errors in the N + 1-dimensional space of all combinations of weights and displacements with a total number of N. The goal of learning in geometric analysis or studying the surface of errors is to find a global minimum on it. In the case of a linear network model and minimizing the sum of squared errors, the error surface is a paraboloid that has a single minimum point. In the case of a nonlinear model, the error surface has a more complex structure and can have local minima, flat areas, saddle points, and long, narrow ravines. Typically, when training such a network, the gradient of the error function is calculated at a randomly selected point on the surface, and then this information is used to move down the slope. The promotion algorithm terminates at the minimum point, either local or global. In essence, algorithms for learning neural networks are similar to algorithms for finding the global extremum of a function of many variables.

Networks with a large number of weights allow you to reproduce complex functions, but in this case the so-called "overtraining" of the network is possible, when the training errors are small, but the resulting model has a weak relationship to the true dependence. A neural network with a small number of weights may not

be flexible enough to model the existing dependence. To overcome the effect of retraining, a control check mechanism is used. Some of the training observations are reserved as control observations and are not used in network training. If the control error stops decreasing or starts to increase, it means that the network is following the input data too closely and training should be stopped. In this case, the number of neurons or layers should be reduced, as the network is too powerful for the problem being solved.

The ability of a network trained on some set of data to produce correct results for a fairly wide class of new data, including those not presented during training, is called the generalization property of a neural network.

To adjust the parameters of neural networks, the adaptation procedure is also widely used, when weights and biases are selected using arbitrary functions of their adjustment, ensuring correspondence between the inputs and the desired values at the output.

Methods for determining the extremum of a function of several variables are divided into three categories - methods of zero, first and second order:

– zero-order methods, in which only information about the values of the function at given points is used to find the extremum;

– first-order methods, where the gradient of the functional error by parameters is used to find the extremum;

– second-order methods that calculate the matrix of the second derivatives of the error functional (the Hessian matrix).

In addition to the listed methods, stochastic optimization algorithms and global optimization algorithms that iterate over the values of the arguments of the error function can also be distinguished.

The method of backpropagation of the error was proposed by several authors independently in 1986 for multilayer networks with direct signal transmission. A

multi-layer neural network is able to perform any display of input vectors on weekends (Fig. 2.18).

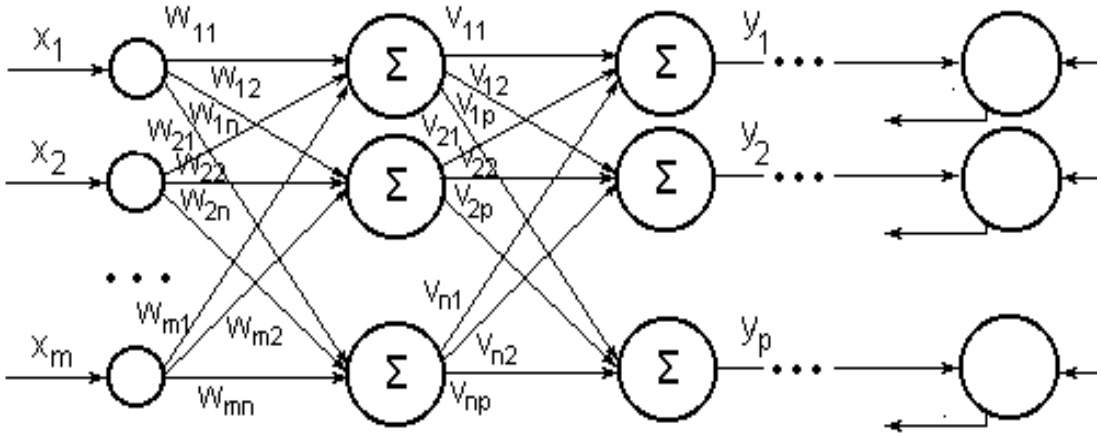

Figure 2.18 – A multilayer network with a hidden layer of n neurons

The input layer of the neural network performs distribution functions. The output layer of neurons serves to process information from previous layers and output results. The layers of neural elements located between the input and output layers are called hidden. Like the original layer, hidden layers are editable. The output of each neuron of the previous layer of the network is connected by synaptic connections to all the inputs of the neural elements of the next layer. As an activation function in multilayer networks, the logistic function and hyperbolic tangent are used more often than others. The error backpropagation method minimizes the root mean square error of the neural network, while the gradient descent method is used in the space of neural network weights and shifts.

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \cdot \frac{\partial E(k)}{\partial w_{ij}(t)}, \tag{2.17}$$

$$S_{0j}(t+1) = S_{0j}(t) - \alpha \cdot \frac{\partial E(k)}{\partial S_{0j}(t)}, \text{ i=1,2,...,n; j=1,2,...,p,} \tag{2.18}$$

where E is the root mean square error of the network for one of the k images, determined by the formula

$$E = \frac{1}{2}\sum_{j=1}^{p}(y_j - t_j)^2, \tag{2.19}$$

where tj is the desired or target output value of the jth neuron.

The error of the jth neuron of the output layer is equal to:γj = yj – tj.

For any hidden layer, the error of the ith neuron element is determined recursively through the errors of the neurons of the next layer j:

$$\gamma_i = \sum_{j=1}^{m}\gamma_j \cdot F'(S_j) \cdot w_{ij}, \tag{2.20}$$

where m is the number of neurons of the next layer in relation to layer i,

wij– weight or synaptic connection between i-th and j-th neurons of different layers,

*Sj*is the weighted sum of the jth neuron.

The weighting coefficients and displacement of neural elements change over time as follows:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \cdot \gamma_j \cdot F'(S_j) \cdot y_i, \tag{2.21}$$

$$S_{0j}(t+1) = S_{0j}(t) + \alpha \cdot \gamma_j \cdot F'(S_j), \qquad \text{i=1,2,...,n, j=1,2,...,p,}$$

whereα- speed of learning.

This rule is called the generalized delta rule. For the logistic activation function:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \cdot \gamma_j \cdot y_j(1-y_j) \cdot y_i \qquad (2.22)$$
$$S_{0j}(t+1) = S_{0j}(t) + \alpha \cdot \gamma_j \cdot y_j(1-y_j),\ \text{i=1,2,...,n, j=1,2,...,p,}$$

the error of the jth neuron of the output layer is defined as

$$\gamma_J = yj - tj, \qquad (2.23)$$

and the jth neuron of the hidden layer

$$\gamma_j = \sum_{i=1}^{m} \gamma_j \cdot y_i \cdot (1-y_i) \cdot w_{ij}, \qquad (2.24)$$

where m is the number of neurons of the next layer in relation to layer j.

Disadvantages of the backpropagation method include the following:

– slow convergence of the gradient method with a constant learning step;

– mixing of local and global minimum points is possible;

– the effect of random initialization of weighting coefficients on the speed of finding a minimum.

To overcome them, several modifications of the backpropagation algorithm are proposed:

1) with an impulse that allows you to take into account the current and previous gradients (heavy ball method), the weight change then:

$$\Delta w_{ij}(t+1) = \alpha \cdot \gamma_j \cdot F'(S_j) \cdot y_i + \eta \cdot \Delta w_{ij}(t), \qquad (2.25)$$

whereα– learning rate coefficient,

η– impulse or moment, usually$0<\alpha<1, \eta\approx 0.9$;

2) with an adaptive learning step that varies according to the formula:

$$\alpha(t) = \frac{1}{1 + \sum x_i^2(t)}; \tag{2.26}$$

3) with a modification according to Rosenberg, proposed by him to solve the problem of converting English printed text into a quality language:

$$\Delta w_{ij}(t+1) = (1-\alpha)\gamma_j \cdot F'(S_j) \cdot y_i + \alpha \cdot \Delta w_{ij}(t), \tag{2.27}$$
$$w_{ij}(t+1) = w_{ij}(t) + \eta \cdot \Delta w_{ij}(t+1).$$

The use of second-order derivatives to correct the weights of the backpropagation algorithm, as practice has shown, did not significantly improve solutions to applied problems.

Procedures related to setup and learning by the error backpropagation method with impulse and adaptive step learning are named learngdx and traingdx respectively in the NNT package.

For multilayer forward signal transmission networks with a logistic activation function, it is recommended to initialize the random initial values of the weights according to the Palmer rule

$$w_{ij} \approx \frac{1}{\sqrt{n(i)}}, \tag{2.28}$$

where n(i) is the number of neural elements in layer i.

According to other authors, the initial weight coefficients should be chosen in the range [-0.05; 0.05] or [-0.1; 0.1]. At the same time, the displacement S0 takes unit values at the initial moment of time. In addition, the number of neural elements of the hidden layers should be less than the training images. To ensure the necessary generalization ability of the network, it is possible to use a network with several hidden layers, the dimension of which is smaller than for a network with one hidden layer. However, neural networks with several hidden layers learn much more slowly.

Recurrent and recirculating neural networks belong to this class of networks. In recirculation networks, information is distributed over bidirectional links that have different weights in different directions. Learning recirculation networks is carried out without a teacher. In contrast to them, recurrent neural networks are characterized by learning with a teacher and feedback, which transmits the results of data processing by the network at the previous stage. The training of such networks is based on the backpropagation algorithm, so they belong to the same class.

Recurrent neural networks are used to solve forecasting and management tasks. In 1986, Jordan proposed the architecture of a recurrent network, in which the outputs of the elements of the last layer are connected to the neurons of the intermediate layer with the help of special input neurons, which are called context neurons [99]. The number of context neurons is equal to the number of initial elements of the recurrent network. The activation function of the output layer is a linear function. The weighted sum of the ith element of the intermediate layer then

$$S_i(t) = \sum_{j=1}^{n} w_{ij} \cdot x_j(t) + \sum_{k=1}^{p} w_{ki} \cdot y_k(t-1) - S_{0i}, \qquad (2.29)$$

where wji is the weight coefficient between the j-th neuron of the input layer and the i-th neuron of the intermediate layer,

*wki*is the weight between the k-th context neuron and the i-th neuron of the intermediate layer,

S0i– displacement of the ith neuron of the intermediate layer,

n is the dimension of the input layer,

p is the number of neurons of the output layer.

The output value of the ith neuron of the hidden layer is then determined as follows:

$$yes(t)=F(Si(t)). \quad (2.30)$$

Nonlinear transformation in the network is performed, as a rule, by logistic functions or hyperbolic tangent.

Another version of the recurrent network was proposed by Elman in 1990. The outputs of the neural elements of the intermediate layer of such a network are connected to the context neurons of the input layer. The number of contextual neural elements is equal to the number of neurons of the intermediate layer. Recurrent networks combining these two approaches are also used. The recurrent network learning algorithm includes the following steps:

1. All context neurons are set to the zero state at t = 0.
2. The input image is fed to the network and the signal is directly propagated.
3. The weighting coefficients are modified and shifted according to the error backpropagation algorithm.
4. *t*is increased by one, and if the rms error of the network E> Emin, then a return to step 2 is made.

Recirculation neural networks are characterized by the fact that in the process of learning a neural network, as a rule, three cycles of information propagation are

produced for each input image: direct, reverse and forward. Let xi (0) be the input vector arriving at the input of the network at the initial moment of time. Then the output vector at the time t = 1 is determined as a result of the direct transformation of information:

$$y_j(1) = \sum_{i=1}^{n} w_{ij} x_i(0), .j = \overline{1, n_1} \tag{2.31}$$

The vector resulting from the inverse transformation vector Y (1):

$$\bar{x}_i(2) = \sum_{j=1}^{n_1} w'_{ji} y_j(1), .j = \overline{1, n} \tag{2.32}$$

At the third stage of information dissemination, the vector (3) is determined:$\overline{Y}$

$$\bar{y}_j(3) = \sum_{i=1}^{n} w_{ij} \bar{x}_i(2),\ j = \overline{1, n_1}. \tag{2.33}$$

Such a transformation of information can be imagined in the form of a chain.

Then the information recovery error in the first layer of the neural network is defined as

$$E = \frac{1}{2} \sum_i (\bar{x}_i(2) - x_i(0))^2. \tag{2.34}$$

The information reproduction error in the second layer of the neural network is determined as follows:

$$E' = \frac{1}{2}\sum_j (\bar{y}_j(3) - y_j(1))^2. \quad (2.35)$$

The training of the neural network is carried out both with the aim of minimizing the error E and E. At the same time, the value yj (1) in the expression is taken as a reference. Then, according to the method of gradient descent in the space of weighting coefficients:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \cdot \frac{\partial E'}{\partial w_{ij}(t)}, \quad (2.36)$$

$$w'_{ji}(t+1) = w'_{ji}(t) - \alpha \cdot \frac{\partial E}{\partial w'_{ji}(t)}. \quad (2.37)$$

Let's define derivatives for a linear neural network. Then:

$$\frac{\partial E'}{\partial w_{ij}(t)} = (\bar{y}_j(3) - y_j(1))\bar{x}_i(2), \quad (2.38)$$

$$\frac{\partial E}{\partial w'_{ji}(t)} = (\bar{x}_i(2) - x_i(0))y_j(1). \quad (2.39)$$

As a result, the expression for setting the weights of the neural network will take the following form:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \cdot \bar{x}_i(2) \cdot (\bar{y}_j(3) - y_j(1)), \quad (2.40)$$

$$w'_{ji}(t+1) = w'_{ji}(t) - \alpha \cdot y_j(1) \cdot (\bar{x}_i(2) - x_i(0)). \quad (2.41)$$

To obtain orthonormalized weight vectors wk for each neuron, it is necessary to introduce a normalized learning rule. LetWk=(w1k,w2k,…,wnk)is the weight vector of the k-th neural element. Then its length at time t + 1 is equal to

$$\left|W_k(t+1)\right| = \sqrt{w_{1k}^2(t+1) + w_{2k}^2(t+1) + \ldots + w_{nk}^2(t+1)}. \qquad (2.42)$$

Accordingly, the normalized learning rule for the weights of the k-th neuron can be represented as follows:

$$w_{\text{ik}}(t+1) = \frac{w_{\text{ik}}(t) - \alpha \cdot \overline{x}_i(2) \cdot (\overline{y}_k(3) - y_k(1))}{|W_k(t+1)|}. \qquad (2.43)$$

Similarly, the formation of weight coefficients is carried outW'. Training is carried out until the total root mean square error of the network becomes less than the specified one.

For the cumulative delta rule, it is not easy to choose the appropriate learning stepα, which ensures a quick achievement of the minimum root mean square error. Adaptive step learning can be used to speed up the learning process. Then, when setting the weighting coefficients of the forward signal transmission network, the value of the learning rate can be determined as in:

$$\alpha(w) = \frac{1}{\sum_{i=1}^{n} \overline{x}_i^{\,2}(2)}, \qquad \alpha(w') = \frac{1}{\sum_{j=1}^{n_1} y_j^{\,2}(1)}. \qquad (2.44)$$

The given algorithm is characterized by the instability of the learning process, and to overcome this shortcoming, you can use the algorithm of layer-by-layer

modification of weights, which more adequately reflects the solution of this problem.

Regression analysis is associated with finding an estimate of the value of a continuous numerical output variable based on the values of the input variables. The task of approximating experimental data can also be solved with the help of artificial neural networks of the following types: multilayer perceptron, radial basis networks, generalized regression networks, probabilistic networks. Artificial neural networks represent output values, as a rule, in a certain range or scale. The simplest of the scaling functions is the minimax function, which performs a linear transformation after determining the minimum and maximum values of the function so that the resulting values are in the desired range. The task of approximating a function for a neural network is formulated as a task of controlled learning (learning with a teacher).

The essence of the task is as follows. There are function values at individual points (nodes), a system of basic functions and vectors of adjustable weighting coefficients. It is necessary to train the network - to choose weighting coefficients for the basis functions so that their combination gives a similar dependence that best approximates many values of the response function.

Radial functions that change monotonically, depending on the distance to the central point, have the form:

Gaussian - $F(S) = \exp\left(-\frac{S^2}{2\sigma^2}\right)$;

multiquadratic - $F(S) = \sqrt{S^2 + \sigma^2}$;

inverse multiquadratic – $F(S) = \frac{1}{\sqrt{S^2 + \sigma^2}}$.

Parameter σ determines the radius of influence of each basis function and the speed of its tendency to zero when moving away from the center. These functions

can be applied to approximate data in multidimensional space. The systems of linear equations to which they lead have a unique solution. The center point, distance scale, and shape of the radial function are model parameters and are completely fixed if the model is linear. A Radial Basis Network (RBF) will be nonlinear if the basis function can shift or change size, or if it contains more than one hidden layer. Usually, the RBF network includes an input layer, a hidden layer consisting of radial elements, and a linear output layer.

Generalized regression and probabilistic networks as wellhave radial base layers with the number of neurons, which is equal to the number of elements or less than the training set, but unlike the usual RBF network, they also include linear and competing layers, respectively. For the generalized regression network, a target array is selected as an initial approximation of the weight matrix of the second linear layer, and a vector corresponding to the average of several target vectors associated with the input vectors close to the given input vector is formed at the output. Probabilistic networks are usually used to solve classification problems. In the competing layer of such networks, the probabilities of the input vector belonging to one or another class are compared and, ultimately, the vector of the class with the higher probability of belonging is determined.

Neural models of networks, including RBF networks, are non-parametric models and their weights (and other parameters) do not have a certain meaning in relation to the problem in which they are applied. The input of the activation function is defined as the modulus of the difference between the weights vector (W) and the input vector (P) multiplied by the offset (S0). The radial basis function used in the NNT package is:

$$\text{Radbas}(S)=\exp(-(|WP)|\cdot S0)2). \qquad (2.45)$$

This function has a maximum of one when the input is 0. As the distance between the vectors W and P decreases, the output of the radial basis function increases. Thus, the radial basis neuron acts as an indicator that generates a value of 1 when the input P is identical to the weight vector W. The shift S0 allows the sensitivity of the radial basis neuron to be adjusted.

After the input vector is assigned, each hidden layer neuron finds the output value according to how close the input vector is to the weight vector of each neuron. Radial basis neurons with weight vectors significantly different from the input vector P will have outputs close to 0 and will have little effect on the outputs of linear neurons.

The amount of influence or draft (spread) or the smoothing coefficient of the radial basis function determines the width of the "caps" of the Gaussian functions with the center in each training observation. A small value of draft leads to a function with sharp peaks and a small approximation error, but such a network is not capable of generalization and can poorly approximate the observation of the control set.

The process of learning radial basis networks includes two stages: the process of setting the centers of basis functions and training neurons in the hidden layer, so RBF networks learn quite quickly. First, the centers of the basis functions are adjusted, then the output layer is trained with fixed parameters of RBF neurons. An alternative technique is to obtain a large number of input images, select possible center values based on them, and then use conventional learning methods to adjust the centers and weights.

Conclusions to section 2.

Various architectures of neural networks and activation functions, their advantages and disadvantages are considered. The classification of neural networks and features of learning neural networks with different architectures are considered.

3 DEVELOPMENT OF SOFTWARE FOR STATISTICAL DATA PROCESSING

3.1 Output data

On March 3, 2020, the first confirmed case of infection with the COVID-19 coronavirus was registered in Ukraine, detected in the Chernivtsi region. On March 25, the Cabinet of Ministers of Ukraine imposed a state of emergency on the entire territory of Ukraine for the period until April 24, which was extended until May 11. On May 4, the quarantine was extended until May 22, with simultaneous mitigation of the consequences on May 11. On November 16, 2020, the prime minister called the collapse of the medical system the main goal. However, information appeared in the mass media about the death of people from the coronavirus as a result of not providing medical assistance. In particular, we are talking about the lack of oxygen concentrators in hospitals, the refusal of hospitals to hospitalize people with severe symptoms, the lack of places and even the dead lying in the wards for hours, etc. According to the data of the Prozorro system, as of November 16, 2020, 9 thousand were purchased since January. Oxygen concentrators for UAH 503 million. As of July 2021, two waves of disease have passed in Ukraine, 2.25 million cases have been registered, of which more than 52 thousand ended in death.

The simulation model of the dynamics of the epidemic process of COVID-19 is confirmed by the data on the incidence of COVID-19, provided by the Resource Center for Coronaviruses of the Johns Hopkins University for Great Britain, Germany and Japan, as well as by the Center for Public Health of the Ministry of Health of Ukraine for Ukraine. The software implementation of the model is implemented in the Python programming language.

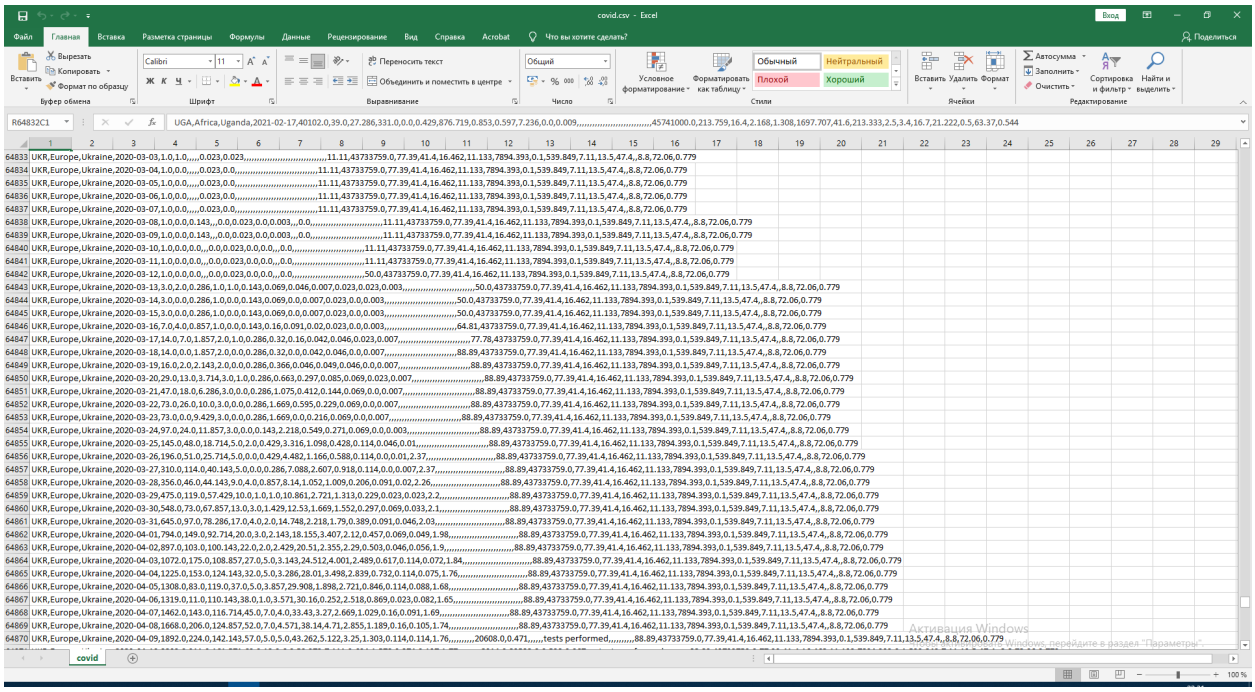

Figure 3.1 – Set of raw data

3.2 Designing business processes of the system

The most important stage in the creation of any information system is the design of a tool capable of implementing all the tasks set at the beginning of the project. The graphically presented scheme of work performance, information exchange, and document flow visualizes the business process model. A graphic presentation of this information allows you to transfer the task of managing the organization from the field of complex craft to the field of engineering technologies.

IDEF0 and DFD methodologies were used in this context. IDEF0 is a functional modeling methodology and graphic notation intended for the formalization and description of business processes, the main difference of this notation is that it does not consider temporal relationships between processes and jobs, but logical relationships (Fig. 3.1-3.2).

Let's consider in more detail the architecture of the project, approaches to solving the tasks and mechanisms of their implementation.

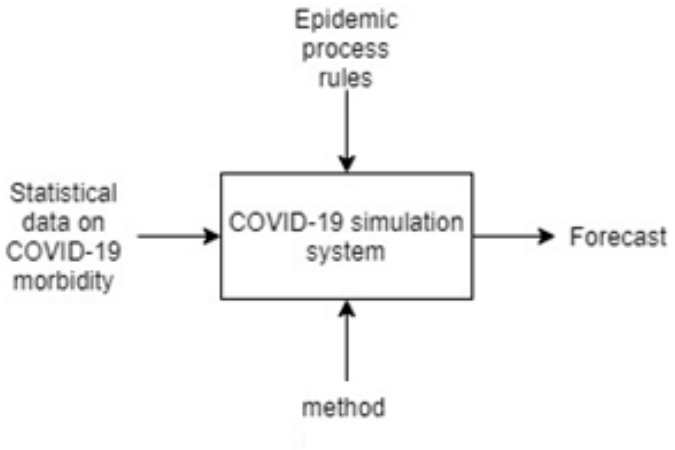

Figure 3.1 – Functional model of the system

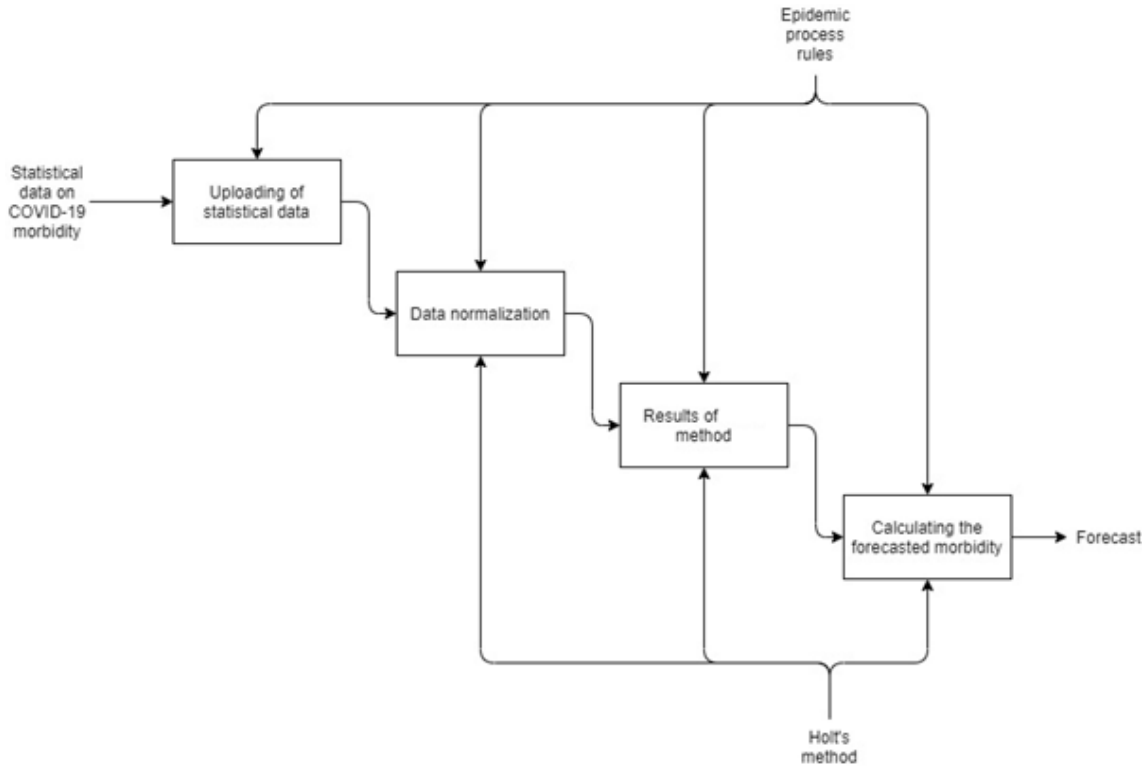

Figure 3.2 - Decomposition of the system

Data flow diagram (DFD) is one of the main tools of structural analysis and design of information systems

(Fig. 3.3), which existed before the widespread use of UML. Despite the fact that there is a shift in emphasis from a structural to an object-oriented approach to the analysis and design of systems in modern conditions, "ancient" structural notations are still widely and effectively used both in business analysis and in the analysis of information systems.

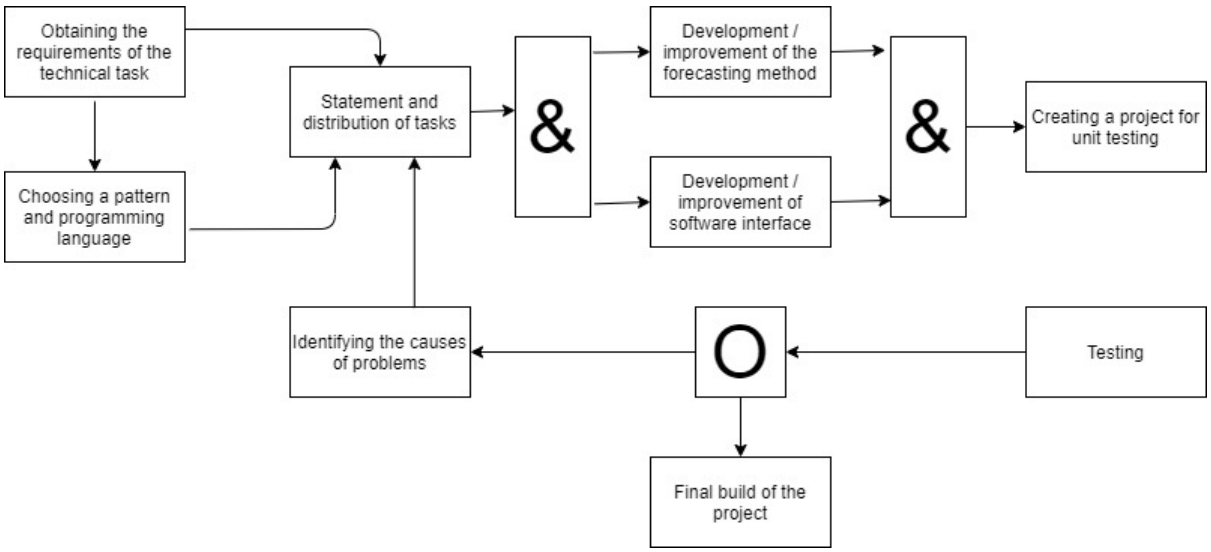

Figure 3.3 – IDEF3 system model

3.3 Object-oriented system design in the UML language

UML (Unified Modeling Language) is a graphical description language for object modeling in the field of software development, business process modeling, system design, and mapping of organizational structures.

3.3.1. State diagrams.A diagram of states is a diagram that defines the change of states of an object over time, one of the behavior modeling diagrams in UML. Represents the object as an automaton from the theory of automata with standardized notation.

In fig. 3.4–3.5 you can see the state diagrams for the prediction system using

a neural network.

Figure 3.4 – General state diagram of the system

The elements of the diagram are:

– A circle representing the initial state.

– A circle with a small circle inside, representing the final state (if any).

– A rounded rectangle that represents a separate state. The top of the rectangle

contains the name of the state, in the middle there can be a horizontal line under which the activities occurring in this state are recorded.

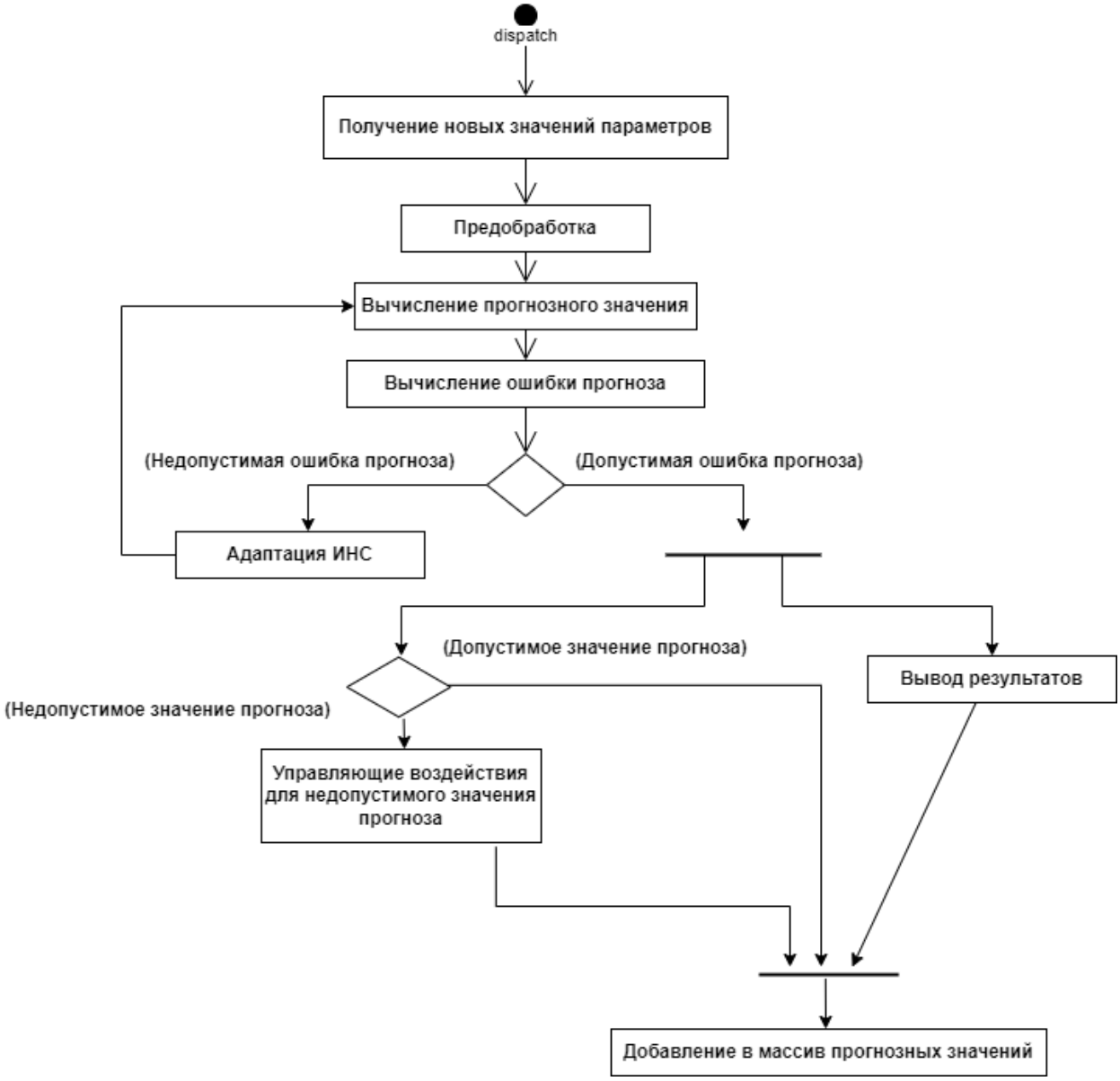

Figure 3.4 – State diagram of the artificial neural network learning process system

– An arrow indicating a transition. The name of the event (if any) that causes the transition is marked above/below the arrow. A guard expression can be added before "/" and enclosed in square brackets (event_name), it means that the transition occurs only if the expression is true. If some activity occurs during the transition, it is added after "/" (event name).

– A thick horizontal line that is the point of joining or branching transitions.

3.3.2 Class diagram. A class diagram defines the types of system classes and the various static relationships that exist between them. Class diagrams also show class attributes, class operations, and restrictions on relationships between classes. An attribute is an element of information associated with a class.

Since attributes are contained within a class, they are hidden from other classes. In this regard, it may be necessary to specify which classes have the right to read and change attributes. This property is called attribute visibility. Four possible values of this parameter can be defined in the attribute:

– public (general, open). This visibility value assumes that the attribute will be visible to all other classes. Any class can view or change the value of an attribute. According to the UML notation, a common attribute is preceded by a "+" sign;

– private (closed, secret). The corresponding attribute is not visible to any other class. A closed attribute is denoted by the sign “–” according to the UML notation;

– protected (protected). This attribute is available only to the class itself and its descendants. The UML notation for a protected attribute is the "#" sign.

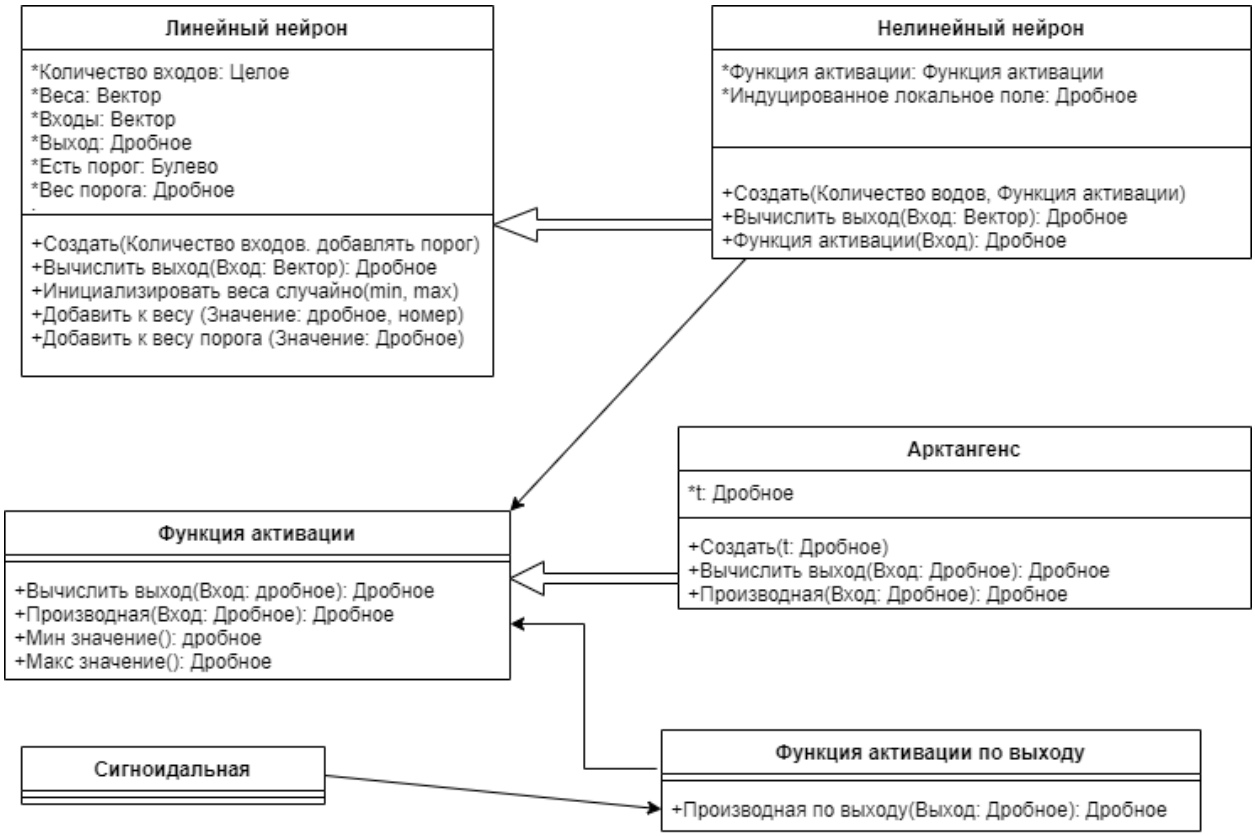

Figure 3.9 – Class diagram

Conclusions to section 3

The chapter described input data to the software product and high-level modeling in terms of IDEF0, DFD and UML diagram methodologies.

4 FEATURES OF THE SOFTWARE IMPLEMENTATION OF ARTIFICIAL NEURAL NETWORKS FOR THE STUDY OF THE EPIDEMIC PROCESS OF COVID-19

4.1 Features of software implementation

Python version 3.3 and higher with the framework installed is required for use:

– Dash, which allows Python to create classes for all visual components of an application. Developers are provided with a set of components in the so-called dash_core_components and dash_html_components. But it also has the ability to build a custom component using JavaScript and React.js.

– Plotly is an open source library for creating interactive graphs and charts, built on plotly.js, which in turn is based on d3.js, which simplifies work with dataframes.

– NumPy is a Python library that adds support for large multidimensional arrays and matrices, along with a large library of high-level (and very fast) math functions for manipulating these arrays.

The process of learning the developed network configuration is carried out iteratively, according to the algorithm of backpropagation of the error. At the first stage of each iteration, the data of the next training example enters the neurons of the input layer and propagates from the first layer to the last, while the output value of each neuron is calculated according to formula (1):

$$OUT_q = f_a(\sum_{p=1}^{N} OUT_p w_{pq}), \tag{4.46}$$

where OUT_q , OUT_P are the output values of neurons q and p, respectively,

f_a – activation function,

w_{pq} is the weighting coefficient of the connection between neurons p and q.

At the second stage of the learning iteration, the weight coefficients of neural connections are recalculated according to the formula (4.47). The calculation is carried out starting from the last layer and ending with the first:

$$w_{pq}(i+1) = w_{pq}(i) + n\delta_q OUT_p, \tag{4.47}$$

where $w_{pq}(i+1)$ – the new value of the weight coefficient of the connection between neurons p and q;

w_{pq} – the old value of the pq-connection weighting factor;

n – learning rate;

δ_q – delta coefficient of neuron q;

OUT_p is the output value of neuron p.

The delta coefficient, which is involved in the calculation of weight values, is calculated for the original layer according to formula (4.48), and for hidden layers according to formula (4.49):

$$\delta_q = OUT_q(1-OUT_q)(ист.зн._q - OUT_q), \tag{4.48}$$

$$\delta_q = OUT_q(1-OUT_q)\sum_{p=1}^{N} OUT_p w_{pq}, \tag{4.49}$$

where OUT_q , OUT_p – output values of neurons q and p, respectively;

w_{pq} is the weighting coefficient of the connection between neurons p and q.

The Dense layer implements the operation: output = activation (dot (input, kernel) + bias), where activation is the element-wise activation function passed as the activation argument, Kernel is the weight matrix created by the layer, and bias is the bias vector, created by the layer (applies only if use_bias is True). If the input to the layer has a rank greater than 2, then it is smoothed before outputting from the Kernel. The arguments are:

– units: a natural number, the dimension of the original space.

– activation: the activation function used. If nothing is specified, no activation is applied (ie "linear" activation: a (x) = x).

– use_bias: Boolean whether the layer uses a bias vector.

– kernel_initializer: initializer for the Kernel weight matrix.

– bias_initializer: initializer for the bias vector.

– kernel_regularizer: regularizer function applied to the Kernel weight matrix.

– bias_regularizer: The regularizer function is applied to the bias vector.

– activity_regularized: the regularizer function is applied to the output of the layer (its "activation").

– kernel_constraint: the constraint function applied to the kernel weight matrix (see constraints).

– bias_constraint: Constraint function applied to the bias vector.

The Dropout layer consists of arbitrarily setting a fraction of the input units to 0 at each update during training, which helps prevent overfitting. The arguments are:

– rate : floats between 0 and 1. The proportion of input blocks to exclude.

– noise_shape : 1D integer tensor representing the shape of the binary dropout mask that will be multiplied by the input.

– seed : Python integer to use as a random seed.

Flatten layer, keras.layers.Flatten (data_format = None) – flattens the input. The argument is data_format : – a string, one of channel_last (default) or

channels_first – the order of passing sizes at the inputs. The purpose of this argument is to preserve the order of weights when switching the model from one data format to another. channels_last matches inputs of the form (packet, ..., channels), and channels_first matches inputs of the form (packet, channels, ...).

The Input layer, keras.engine.input_layer.Input(), is used to initialize the Keras tensor. A Keras tensor is a tensor object from a substrate (Theano, TensorFlow, or CNTK), which we supplement with certain attributes that allow us to build a Keras model simply by knowing the inputs and outputs of the model. His arguments:

- shape : shape tuple (integer), not including batch size. For example, shape = (32,) indicates that the expected input will be batches of 32-dimensional vectors.
- batch_shape : shape tuple (integer) including batch size. For example, batch_shape = (10, 32) indicates that the expected input will be batches of 10 32-dimensional vectors. batch_shape = (None, 32) specifies that batches with any number of 32-dimensional vectors.
- name : Optional name string for the layer. Must be unique in the model. It will be automatically generated if not provided.
- dtype : data type expected on input, as a string (float32, float64, int32 ...)
- sparse : Boolean function indicating whether the holder being created is sparse.
- tensor : an additional existing tensor to wrap in the input layer.

The Reshape layer, keras.layers.Reshape (target_shape), reforms the output into a specific shape. The argument is target_shape : – the target shape. A tuple of integers. The input form is optional, although all dimensions in the input form must be fixed. The input_shape key argument (a tuple of integers, not including the reference axis) is used when using this layer as the first layer in the model.

Permute layer, keras.layers.Permute (dims), Keeps the size of the input signal according to the given sample. Useful, for example, for connecting RNNs and

convolutions. The argument is dims : a tuple of integers. Does not include sample size. Indexing starts at 1. For example, (2, 1) iterates over the first and second dimensions of the input parameter. The input form is arbitrary. The input_shape key argument (a tuple of integers, not including the reference axis) is used when using this layer as the first layer in the model.

ActivityRegularization layer, keras.layers.ActivityRegularization (l1 = 0.0, l2 = 0.0) – A layer that applies an update to the input data based on a cost function. The input_shape key argument (a tuple of integers, not including the reference axis) is used when using this layer as the first layer in the model.

Activations can be used both through the activations layer and through the activations argument supported by all previous layers. The following activation functions are available:

– Elu is an exponential linear block that returns an exponential linear activation function: x if x> 0 and alpha * (exp (x) -1) if x < 0.

– Softmax – Softmax activation function, returns a tensor as the output of a softmax transform.

– Selu – Scaled Exponential Linear Unit (SELU). SELU is equal to: scale * elu (x, alpha), where apha and scale are conditional constants. The values of alpha and scale are chosen so that the mean and variance of the inputs are preserved between two successive layers as long as the weights are properly initialized (see Lecun_normal initialization) and the number of inputs is "large enough" (see Reference for more information). Returns the resized exponential activation function: scale * elu (x, alpha).

– Softplus – Softplus activation function. Returns the Softplus activation function: log (exp (x) + 1).

– Softsign – softsign activation function. Returns the softsign activation function: x / (abs (x) + 1).

– Relu - rectification, linear block. With default values, it returns by elements. max (x , 0). Otherwise: f (x) = max_value for x> = max_value, f (x) = x for threshold <= x < max_value, f (x) = alpha * (x - threshold).

– Tanh is an activation function in the form of a hyperbolic tangent. Returns the hyperbolic function: tanh (x) = (exp (x) - exp (-x)) / (exp (x) + exp (-x))

– Sigmoid – activation function in the form of a sigmoid. Returns the sigmoid activation function: 1 / (1 + exp (-x)).

– hard_sigmoid function to activate the "hard" sigmoid. It is faster to calculate than the sigmoid activation function.

– Returns: 0 if x <-2.5, 1 if x> 2.5, 0.2 * x + 0.5 if -2.5 <= x <= 2.5.

– Exponential – exponential (base) activation function. Returns the exponential activation function: exp (x).

– Linear – function of linear (i.e. identification) activation. Returns the input tensor, unchanged.

A metric is a function used to evaluate the performance of your model. Metric functions are provided in the metrics parameter when compiling the model. The metric function is similar to the loss function, except that the results of the metric evaluation are not used when training the model. Any of the loss functions can be used as a metric function. Available metrics: accuracy, binary_accuracy, categorical_accuracy, sparse_categorical_accuracy, top_k_categorical_accuracy, sparse_top_k_categorical_accuracy, cosine_proximity, clone_metric.

The optimizer is one of two arguments required to compile a Keras model. It is possible to either instantiate the optimizer before passing it to model.compile(), or call it by name. In the latter case, the default optimizer parameters will be used. The following optimizers are available:

– SGD is a stochastic gradient descent optimizer. Includes momentum support, learning rate decay, and Nesterov's momentum.

– RMSprop – RMSProp optimizer. It is recommended to leave the parameters of this optimizer at their default values (except for the learning rate, which can be freely configured).

– Adagrad is an optimizer where the learning rate depends on specific parameters that are tailored to how often the parameter is updated during training. The more updates a parameter receives, the lower the learning rate. It is recommended to leave the parameters of this optimizer at their default values.

– Adadelta is a more robust extension of Adagrad that adapts the learning rate based on a sliding gradient update window instead of accumulating all the gradients from previous years. Thus, Adadelta continues to learn even when many updates are made. Compared to Adagrad, in the original version of Adadelta there is no need to set the initial learning rate. In this version, as in most other Keras optimizers, you can set the initial learning rate and the decay rate. It is recommended to leave the parameters of this optimizer at their default values.

– Adam is the Adam optimizer. The default parameters correspond to the parameters specified in the original paper.

– Adamax is a variant of Adam based on the infinity norm. The default parameters correspond to the parameters given in the article.

Nadam – Nesterov Adam optimizer. Just as Adam is essentially an RMSprop with momentum, so Nadam is an RMSprop with Nesterov momentum. The default parameters correspond to the parameters given in the article. It is recommended to leave the parameters of this optimizer at their default values.

4.2 Neural network model of the epidemic process of COVID-19

The development and implementation of intelligent management systems for technological, economic, and social processes should include the implementation of

predictive models that allow to significantly improve the quality of management decisions. Getting a forecast in time and increasing its accuracy increase the effectiveness of decision-making, prevent catastrophic situations, and reduce the risks of adverse consequences.

Currently, one of the main directions of research in the field of artificial intelligence systems is neural network technologies. Interest in studying the application of neural networks in forecasting tasks is due to their ability to solve tasks that do not lend themselves to strict formalization, to identify internal, hidden patterns, and to conduct in-depth data analysis.

Building a neural network system includes input data processing, architecture development, and network training. There is no general implementation algorithm for each of the listed stages - the system configuration depends on many factors that are covered by a specific task. Thus, in order to obtain a forecast when developing a neural network, the nature of the forecasted time series, the desired form of obtaining the forecast, the horizon of forecasting, the requirement for the time of obtaining the forecast, and the amount of input data are taken into account. The flexibility and lack of strict formalization in system development provide a wide range of opportunities for research, improvement and adaptation of existing models of neural networks in order to increase the accuracy of the forecast.

When solving the problem of forecasting, the neural network system is built in the following way: the input layer contains several neurons to which the values of the studied time series are fed, and the last layer consists of a single neuron, at the output of which the forecast is obtained.

The disadvantage of the implementation of this algorithm is the fairly rapid accumulation of errors.

As a result of the conducted research, a solution was developed and tested, which aims to eliminate the above-described deficiency in order to increase the

accuracy of the forecast. According to the obtained results, it is proposed to make the following changes in the architecture of the predictive neural network:

– increase the number of neurons of the output layer, which is determined by the number of prediction steps;

– introduce communication between the neurons of the output layer.

The accuracy of the forecast is increased by connecting the output neurons to each other so that the value obtained on the first output neuron is fed to the input of the second output neuron, and the value obtained on the first and second is taken into account on the third, etc. d. In other words, each subsequent neuron of the output layer, in addition to signals from the neurons of the penultimate layer, must be supplied with signals already received at the previous outputs of the network.

To launch the computer program, the file "dash_dark.py" is used, which calls all the necessary modules. The result of the program is recorded in the form of a model file "model.h5", a file "predicted.json" (where current data and data with predictions are stored - Fig. 4.1).

The graphical display can be scaled, a dotted line can be drawn where the model training data ends, and more.

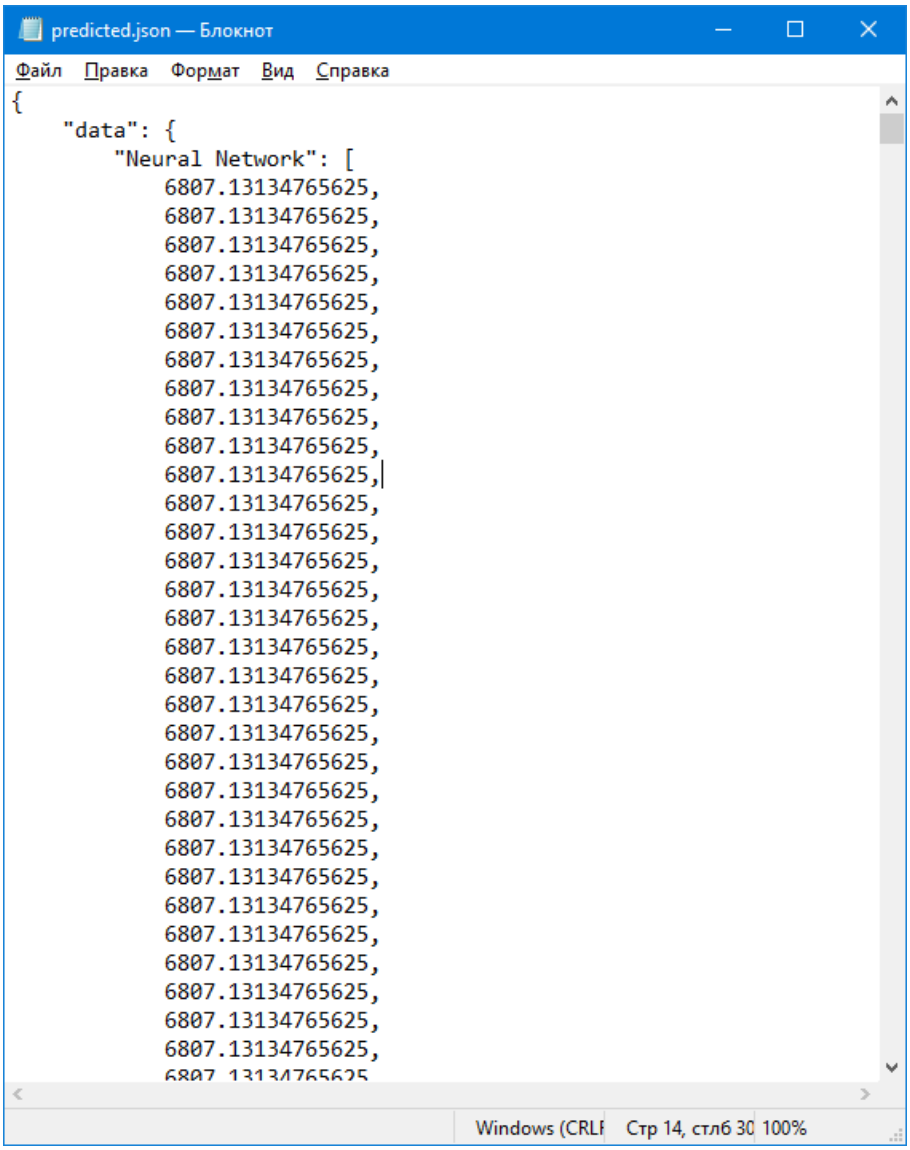

Figure 4.1 – File structure of the "predicted.json" file of the computer program

To predict COVID19, the following neural network was built with the structure: 6 pairs of Dense (relu activation, 64 neurons) and Dropout layers, the last layer is Dense (relu activation), one output, the optimizer is rmsprop, the error function is mse.

To test the adequacy, the model was tested on data on the incidence of COVID-19 in Ukraine.

Figure 4.2 presents the results of predicting the incidence of COVID-19 based on the neural network model.

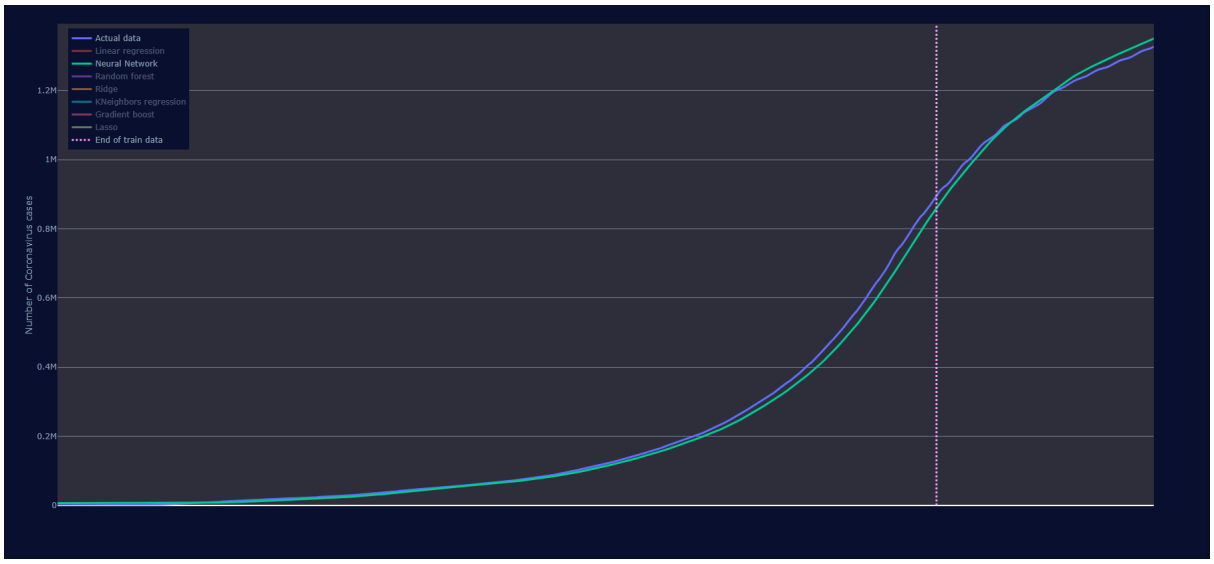

Figure 4.2 – Results of forecasting the incidence of COVID-19 in Ukraine

An important stage of forecasting is the verification of forecasts, that is, the assessment of their accuracy and validity. At the verification stage, a set of criteria, methods and procedures are used to assess the quality of the forecast.

The most common retrospective assessment of the forecast, that is, the assessment of the forecast for the past time.

For this, the source information is divided into two parts, one of which covers earlier data, and the other - more recent. Using the data of the first group (retrospection), the parameters of the forecast model are estimated, and the data of the second group are considered as the actual data of the forecasted indicator. The forecast error obtained retrospectively to some extent characterizes the accuracy of the used forecasting technique.

All indicators used to analyze forecast quality can be divided into three groups: absolute, comparative and qualitative.

Indicators that make it possible to quantitatively determine the amount of forecast error in units of measurement of the forecasted object or in percentages are classified as absolute. These are root mean square error a, absolute error Apr,

average absolute error D|f, relative error epr and average relative error of forecast epr.

The absolute forecast error can be defined as the difference between the actual value (y) and the forecast (y*):

$$\Delta_{\text{пр}} = y_t - y^*.$$

The average absolute value of the error will be:

$$\overline{\Delta_{\text{пр}}} = \frac{\sum_{t=1}^{n} |y_t - y_t^*|}{n} = 6792.95.$$

The root mean square error of the forecast is calculated by the formula

$$\sigma_t = \sqrt{\frac{\sum_{t=1}^{n} (y_t - y_t^*)^2}{n}} = 8292.85.$$

It should be noted that for a large class of statistical distributions there is a relationship between the average absolute deviation D|ir and the standard deviation a, which can be represented in the following form:

$$\sigma_t = 1{,}25\ \overline{\Delta_{\text{пр}}}.$$

The disadvantage of the considered indicators is that the value of these characteristics significantly depends on the scale of measurement of the levels of the studied phenomena.

The absolute error of the Lpr forecast can be expressed as a percentage of the actual values of the indicator in the following way:

$$\varepsilon_{\text{пр}} = \frac{y_t - y_t^*}{y_t} 100,$$

and the average relative error (error of approximation) is calculated as

$$\overline{\varepsilon_{\text{пр}}} = \frac{\sum_{t=1}^{n} \frac{|y_t - y_t^*|}{y_t} 100}{n} = 8{,}941.$$

This indicator, as a rule, is used when comparing the accuracy of forecasts of different forecasting objects. Typical values of EPR for medium-term forecasts and their interpretation are given in the table. 14.10.

Data forecasting and interpretation

- $\overline{\varepsilon_{\text{пр}}} < 10 -$ висока точність*;*
- $\overline{\varepsilon_{\text{пр}}} \in 10 - 20$ – хороша точність*;*
- $\overline{\varepsilon_{\text{пр}}} \in 20 - 50$ – задовільна точність;
- $\overline{\varepsilon_{\text{пр}}} > 50$ – незадовільна точність;

The mean absolute and root mean square errors record the average value of the error at each state of the forecast without taking this error into account. The average error makes it possible to determine which type of error is most typical - underestimation or overestimation of the predicted indicator. It should be borne in mind that Lpr and o are equal to zero only when y and - y* for each £, that is, in the case of a perfect forecast. A similar statement is not valid for the absolute error A, since there may be mutual cancellation of errors. Both the absolute values of the variables and their increments can be used to calculate these indicators.

4.3 Description of the method of creating a model by grid search for the best combination of hyperparameters.

Almost every machine learning algorithm has hyperparameters - parameters whose value depends on the process of learning the model. The process of finding the best hyperparameters is called hyperparameterization.

In order to automate the selection of hyperparameters, you can use the GridSearchCV class. the algorithm used by this class is extremely simple:

- a grid with different values for each hyperparameter is provided;
- a model is trained for each sample from the Cartesian product of sets;
- models are compared with each other using various metrics;
- based on the comparison results, the best model is selected;

Suppose that our algorithm has three hyperparameters - alpha, gamma and n_iter. Let them have the following possible values:

- alpha = [0.1, 0.2, 0.3, 0.4, 0.5]
- gamma = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
- n_iter = [100, 200, 300, 400, 500]

According to the laws of combinatorics, the total number of trained RAVO models is 5 * 10 * 5 = 250. All these models are compared using the evaluation metric (in current case it is MSE - Mean Squared Error) and the best one is selected.

To evaluate the model not only on the training data passed to the fit () method, you should use the PredefinedSplit class and create a pipeline as follows:

```
X, y = data
X_train, X_test, y_train, y_test = train_test_split (X, y)
cv = PredefinedSplit ([- 1 if x in X_train else 0 for x in X])
# Create Pipeline
    pipeline = Pipeline (...)
# Create GridSearch
    grid_search = GridSearchCV (estimator = pipeline, cv = cv, ...)
# Fit model
    grid_search.fit (X, y)
```

```
# Get the best model for the current task
    model = grid_search.best_estimator_
```

Conclusions to section 4

A neural network was built for simulating epidemic processes. The architecture of neural elements was developed, which activation functions should be used, single-layer and multilayer neural networks were considered. A neural network model of the epidemic process of COVID-19 was developed and implemented. The accuracy and adequacy of the model was investigated on the data on the incidence of COVID-19 in Ukraine.

.

CONCLUSIONS

Research in the field of the application of machine learning methods to the diagnosis of diseases of the genitourinary system in children is relevant today and is carried out both in our country and abroad.

Various methods of machine learning and approaches to improving their quality in relation to extracted data were also analyzed. A special software product for processing statistical data of patients was designed and developed.

In the work, solve the following problems:

– An analysis of machine learning methods for studying the epidemic process was carried out.

– Machine learning methods for solving problems of epidemiological diagnostics are classified.

– Developed models of epidemic processes based on machine learning methods.

– The results obtained using the developed machine learning models were studied.

A neural network was built for simulating epidemic processes. The architecture of neural elements was developed, which activation functions should be used, single-layer and multilayer neural networks were considered. A neural network model of the epidemic process of COVID-19 was developed and implemented. The accuracy and adequacy of the model was investigated on the data on the incidence of COVID-19 in Ukraine.

LIST OF USED LITERATURE

1. Alpaydin, E. Machine Learning: The New AI [Text] / E. Alpaydin // MIT Press. - 2016. - 206 p.

2. Lomg, Z. Research on International Relations in the Age of Artificial Intelligence [Text] / Z. Long, J. Chen, T. Bao // 2020 International Conference on Computer Engineering and Application (ICCEA). – 2020. – pp. 436-438, doi: 10.1109/ICCEA50009.2020.00100.

3. Laure, B. Machine Learning to Data Management: A Round Trip [Text] / B. Laure, B. Angela and M. Tova // 2018 IEEE 34th International Conference on Data Engineering (ICDE). - 2018. - pp. 1735-1738, doi: 10.1109/ICDE.2018.00226.

4. Müller, K. Towards robust machine learning methods for the analysis of brain data [Text] / K. Müller // 2018 6th International Conference on Brain-Computer Interface (BCI). - 2018. - pp. 1-2. doi: 10.1109/IWW-BCI.2018.8311495.

5. Pracidelli, LP Fraud identification architecture using data mining and machine learning in a private transport company that operates by applications [Text] / LP Pracidelli, FS Lopes // 2020 15th Iberian Conference on Information Systems and Technologies (CISTI). – 2020. – pp. 1-6, doi: 10.23919/CISTI49556.2020.9140992.

6. Mana, SC A Machine Learning Based Implementation of Product and Service Recommendation Models [Text] / SC Mana, T. Sasipraba // 2021 7th International Conference on Electrical Energy Systems (ICEES). – 2021. – pp. 543-547. doi: 10.1109/ICEES51510.2021.9383732.

7. Bao, Xiao-Hua Modeling and optimization of the claw-pole alternator based on support vector machines and chaos [Text] / Xiao-Hua Bao, Qun-Jing Wang and You-Yuan Ni // 2005 International Conference on Machine Learning and Cybernetics. - 2005. - pp. 4012-4016. doi: 10.1109/ICMLC.2005.1527639.

8. Ramasubramanian, K. Applied Supervised Learning with R: Use machine learning libraries of R to build models that solve business problems and predict future trends [Text] / K. Ramasubramanian, J. Moolayil // Packt Publishing. - 2019. - 502 p.

9. Doroodgar, B. A Learning-Based Semi-Autonomous Controller for Robotic Exploration of Unknown Disaster Scenes While Searching for Victims [Text] / B. Doroodgar, Y. Liu and G. Nejat // IEEE Transactions on Cybernetics, vol. 44, no. 12. – 2014. – pp. 2719-2732. doi: 10.1109/TCYB.2014.2314294.

10. Cunhe, L. A new semi-supervised support vector machine learning algorithm based on active learning [Text] / L. Cunhe, W. Chenggang // 2010 2nd International Conference on Future Computer and Communication. - 2010. - pp. V3-638-V3-641. doi: 10.1109/ICFCC.2010.5497471.

11. Zhang, Z. Learning Automata-Based Multiagent Reinforcement Learning for Optimization of Cooperative Tasks [Text] / Z. Zhang, D. Wang, J. Gao // IEEE Transactions on Neural Networks and Learning Systems. doi: 10.1109/TNNLS.2020.3025711.

12. Hussein, S. Lung and Pancreatic Tumor Characterization in the Deep Learning Era: Novel Supervised and Unsupervised Learning Approaches [Text] / S. Hussein, P. Kandel, CW Bolan, MB Wallace, U. Bagci // IEEE Transactions on Medical Imaging, vol. 38, no. 8. - 2019. - pp. 1777-1787. doi: 10.1109/TMI.2019.2894349.

13. Rakhmanov, O. Testing strength of the state-of-art image classification methods for hand drawn sketches [Text] / O. Rakhmanov // 2019 15th International Conference on Electronics, Computer and Computation (ICECCO). – 2019. – pp. 1-3. doi: 10.1109/ICECCO48375.2019.9043258.

14. Yu, Z. et al. Adaptive Fuzzy Consensus Clustering Framework for Clustering Analysis of Cancer Data [Text] / Z. Yu, et. al. // IEEE/ACM Transactions

on Computational Biology and Bioinformatics, vol. 12, no. 4. – 2015. – pp. 887-901. doi: 10.1109/TCBB.2014.2359433.

15. Cheng, X. Application of regression analysis based on genetic particle swarm algorithm in financial analysis [Text] / Xiaorong Cheng, Lin Sun and Ping Liu // 2010 International Conference On Computer Design and Applications. - 2010. - pp. V4-335-V4-338. doi: 10.1109/ICCDA.2010.5541106.

16. Hosmer, DW Applied Logistic Regression [Text] / DW Hosmer, S. Lemeshow // 2nd ed.. New York; Chichester, Wiley. ISBN 0-471-35632-8.

17. Han, J. Data Mining: Concepts and Techniques [Text] / J. Han, M. Kamber. – Elsevier, 2006.

18. Borgelt, K. Computational Intelligence: A Methodological Introduction [Text] / K. Borgelt, K. Moewes, S. Held // 2013, Springer, ISBN 9781447150121.

19. Christopher M. Bishop Pattern Recognition and Machine Learning [Text] / Hardcover. - 2006. - 740 p.

20. Broomhead, DH Multivariable Functional Interpolation and Adaptive Networks [Text] / DH Broomhead, D. Lowe // Complex Systems: journal, vol. 2. - 1988. - pp. 321—355.

21. Hastie, T. The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations [Text] / T. Hastie // New York: Springer. - 2001. - xvi, 533 p. ISBN 0-387-95284-5, 978-0-387-95284-0.

22. Ayvazyan, S.A. Applied statistics: Classification and dimensionality reduction [Text] / S.A. Ayvazyan, V.M. Bukhstaber, I.S. Enyukov, L.D. Meshalkin // M.: Finances and statistics. - 1989. - 607 p.

23. Hastie, T. The elements of statistical learning: Data mining, inference, and prediction [Text] / T. Hastie, R. Tibshirani, JH Friedman // New York: Springer Verlag.

24. Breiman, L. Classification and regression trees [Text] / L. Breiman, JH Friedman, RA Olshen, CJ Stone // Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software, 1984.

25. Freund, Yoav; Schapire, Robert E (1997). "A decision-theoretic generalization of on-line learning and an application to boosting". Journal of Computer and System Sciences. 55: 119–139.

26. Randal S. Olson, Jason H. Moore Proceedings of the Workshop on Automatic Machine Learning, PMLR 64:66-74, 2016.

27. Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2001). 8.5 The EM algorithm. The Elements of Statistical Learning. New York: Springer. with. 236–243. ISBN 0-387-95284-5

28. Huang, Te-Ming; Kecman, Vojislav; and Kopriva, Ivica (2006); Kernel Based Algorithms for Mining Huge Data Sets, in Supervised, Semi-supervised, and Unsupervised Learning, Springer-Verlag, Berlin, Heidelberg, 260 pp.

29. Rakesh Agrawal and Ramakrishnan Srikant Fast algorithms for mining association rules. Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, pages 487-499, Santiago, Chile, September 1994.