UDK 004-93

N.O. Matveeva

USING NEURAL NETWORKS PROGRAMMING ON JAVA FOR SOLVING THE PROBLEM OF SIGNAL RECOGNITION

Abstract. The results of the study of signal recognition using neural networks are presented. Scanning of composite materials is performed in the presence of additive noise. A multilayer perceptron with back-propagation error is implemented on Java in the environment NetBeans. Experiments were performed to analyze MSE values and accuracy. The confusion matrix, sensitivity and specificity were obtained and analyzed.

Keywords: composite materials, neural networks, multilayer perceptron with back-propagation training, defect, function of activity.

Introduction. A problem of classification is one of most often nascent and decided tasks both in scientific researches, and in practice.

At conducting non-destructive control of composite materials it follows to take difficult relief of surface into account them. The manufacturing techniques of fibrous composites, usually, do not provide for mechanical processing. The surface scanning process is complicated by various types of noise.

When analyzing signals, we obtain information on the presence and size of defects. For the decision of such tasks neural networks are used, what actively develop lately, own universal and adaptive properties and provide high efficiency of recognition[1, 2].

When creating programs in most programming languages, it is necessary to determine the operating system and the type of processor on which they will work. When creating Java applications, you don’t need to think about it. The Java compiler creates an intermediate byte code for the Java Virtual Machine (JVM). The core of the Java virtual machine is practically for all types of computers and operating systems, so the byte-code files are considered independent of the platform [3].

Problem definition. The purpose of this work is to create a neural network using Java language for solving the problem of classification of electromagnetic signals.

 ISSN 1562-9945

Main part. Neural networks can be very well applied to classification problems, where one wants to automatically assign some record to a certain category. This classification tasks falls into the supervised learning category.

A classification algorithm seeks to find the boundaries between classes in the data hyperspace. Once the classification boundaries are defined, a new data point, with an unknown class, receives a class label according to the boundaries defined by the classification algorithm[1].

Classification problems usually deal with a case of multiple classes, where each class is assigned a label. However, a binary classification schema is applied in neural networks. This is because a neural network with a logistic function at the output layer can produce only values between 0 and 1, meaning that it assigns (1) or not (0) to some classes.

There is no perfect classifier algorithm; all of them are subjected to errors and biases. However, it is expected that a classification algorithm can correctly classify 70% to 90% of the records [1].

At conducted to the analysis of entrance and initial data of value of weight and change of neural network automatically put right so that to minimize a difference between a desirable signal and got on an exit as a result of design. This difference is named the error of training and for certain neuron network configuration it is determined by a key-in through the network of all supervisions that is present, and comparison of initial values of , desirable, by a having a special purpose value.

The confusion matrix (criterion of quality of training) is formed. A confusion matrix shows how many of a given class's records were correctly classified and therefore how many were wrongly classified.

When the classification is binary, the confusion matrix is found to be a simple 2x2 matrix, and therefore, its positions are specially named:

Actual Class	Inferred Class	
	Positive(1)	Negative(0)
Positive(1)	True Positive	False Negative
Negative(0)	False Negative	True Positive

For classification of signals the concept of a binary confusion matrix is applied in the sense that a false defect may be either a false positive or a false negative. The rate of false results can be measured by using sensitivity and specificity indexes

Sensitivity denotes the true positive rate; it measures how many of the records are correctly classified positively.

$$Sensitivity=\frac{Number\ of\ True\ Positive}{Total\ of\ Actual\ Positive\ Records} \quad (1)$$

Specificity in turn represents the true negative rate; it indicates the proportion of negative record identification.

$$Specificity = \frac{\textbf{Number of True Negatives}}{\textbf{Total of Actual Negative Records}} \quad (2)$$

High values of both sensitivity and specificity are desired; however, depending on the application field, sensitivity may carry more meaning

A neural network consists of layers, neurons, weights, activation functions, and biases, and there are basically three types of layers: input, hidden, and output. Each layer may have one or more neurons. Each neuron is connected either to a neural input/output or to another neuron, and these connections are known as weights.

It is important to highlight that a neural network may have many hidden layers or none, as the number of neurons in each layer may vary. However, the input and output layers have the same number of neurons as the number of neural inputs/outputs, respectively.

When scanning composite materials using an eddy current transformer, there is a smooth change in the waveform from unimodal with a maximum amplitude (defects exceed the control zone) to bimodal with the highest peaks (point defects). Such changes are modeled using the expression[4]:

$$y(x) = \exp(-1{,}5x^{\uparrow}2) - k \cdot \exp(-3x^{\uparrow}2) \quad (3)$$

The article uses multilayer perceptron (MLP). For the training of MLP the algorithm of reverse distribution of error (back - propagation training) is used [3]. Every neuron of MLP, that training on the basis of reverse distribution has a nonlinear smooth function of activating, as that often use the nonlinear sigmoid function of activating of type of logistic or hyperbolical tangent.

Initially we define six classes: *Neuron, Layer* (class is abstract and cannot be instantiated), *InputLayer* (class inherits attributes and methods from the Layer class), *HiddenLayer* (class inherits attributes and methods from the Layer class), *OutputLayer* (class inherits attributes and methods from the Layer class), *NeuralNet* (the values of the neural net topology are fixed in this class).

Class *NeuralNet* create an MLP with one hidden layer and a sigmoidal function or a hyperbolic tangent at the exit. Each output neuron denotes a class.

Additionally, we add a special *Classification* class to handle such concepts as the discrepancy matrix, sensitivity, and specificity.

The implementation of the neural network for classification will consist of the following steps:

1. Data loading (training and testing data)

2. Data normalization

3. Creation of a neural network

4. Training the neural network

For the decision of the put task multilayer perceptron was used from 21 neurons in input layer (after the amount of components of entrance vector) and 2 neurons in an output layer. The number of neurons in the hidden layer took the values: 4, 8, 9, 11.

NeuralNet n1 = new NeuralNet();

n1 = n1.initNet(21, 1, 11, 2);

Next, we perform the training. Then, we create a new network to receive the trained network:

After the training has been finished, we instantiate a classification object to carry out some analyses on the results: Finally, we apply some processing for exhibiting the charts and the confusion matrix:

The dataset division was performed as follows: for training: 10 records and for test: 8 records

We performed many experiments to try to find the best neural net to classify. We conducted 14 different experiments to analyze the MSE and accuracy values. After that, the confusion matrix, sensitivity, and specificity were generated with the test dataset and analyzed. At last, an analysis of generalization was conducted. The neural networks involved in the experiments are shown in the table 1:

Table 1

Results of experiments with activation functions

Experiment	**Number of neuron in hidden layer**	**Activation function**
1	4	Hidden layer: *hypertan* Output layer: *siglog*
2		Hidden layer: *siglog* Output layer: *siglog*
3	8	Hidden layer: *hypertan* Output layer: *siglog*
4		Hidden layer: *siglog* Output layer: *siglog*
5		Hidden layer: *siglog* Output layer: *hypertan*
6		Hidden layer: *hypertan* Output layer: *hypertan*

7	9	Hidden layer: *hypertan* Output layer: *siglog*
8		Hidden layer: *siglog* Output layer: *siglog*
9		Hidden layer: *siglog* Output layer: *hypertan*
10		Hidden layer: *hypertan* Output layer: *hypertan*
11	11	Hidden layer: *hypertan* Output layer: *siglog*
12		Hidden layer: *siglog* Output layer: *siglog*
13		Hidden layer: *siglog* Output layer: *hypertan*
14		Hidden layer: *hypertan* Output layer: *hypertan*

After each experiment, we collected the MSE values (shown in the tabl. 2); experiment 11 and experiment 12 resulted in the highest accuracy values. Both MSE training rates are acceptable.

Table 2

Experiment	**MSE training rate**	**Accuracy**
1	0.0899686860326715	0.625
2	0.2400169524749906	0.5
3	0.05691983398831299	0.625
4	0.08252472726178053	0.625
5	0.09819078797242933	0.725
6	0.0691983398831299	0.625
7	0.08252472726178053	0.625
8	0.07819078797242933	0.625
9	0.06819078797242933	0.625
10	0.0691983398831299	0.625
11	**5.90576728651045E-4**	**0.625**
12	**3.851055408799272E-4**	**0.875**
13	0.004400628250927104	0.625
14	0.008933135394483038	1.0

Graphically, the MSE evolution over time is very fast, as can be seen in the following chart of the experiment 11:

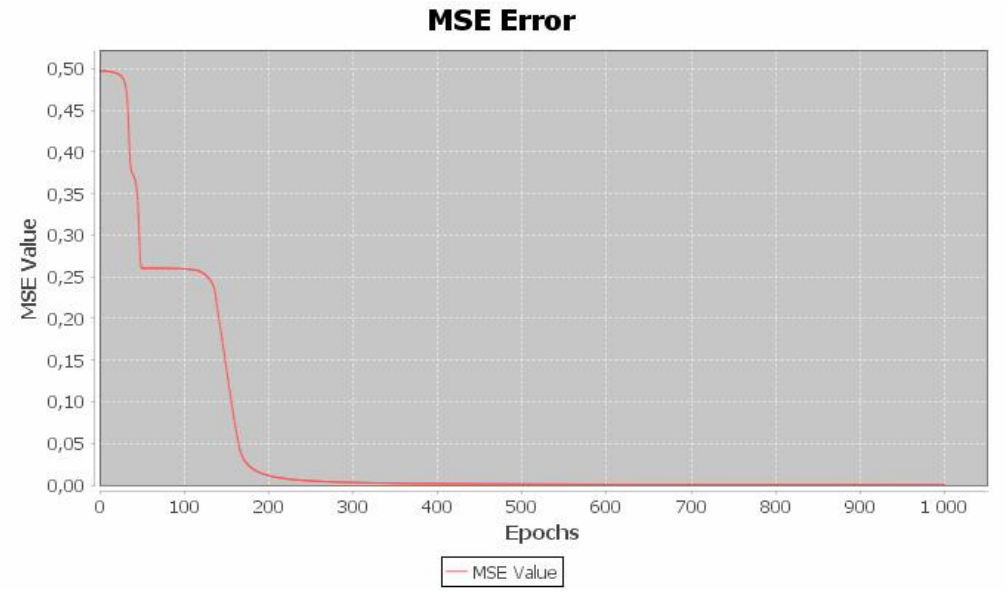

Figure 1 - The values MSE for hidden layer: *hypertan* and output layer: *siglog*

The fall of the MSE is fast; nevertheless, the experiments showed a slight delay in the decrease in the first's epochs.

CONFUSION MATRIX
4.0 | 1.0 |
0.0 | 3.0 |
SENSITIVITY = 1.0
SPECIFICITY = 0.75
ACCURACY = 0.875

Now, let's analyze generalization. This feature is better observed with bar charts showing for each case the expected class along with the classification estimated by the neural network. Red bars denote the actual positive diagnostic, while blue bars represent the neural output values. It is worth to note that when the output is zero, recognition of the point defects, and when the output is one, the defined cracks. This feature is better observed with bar charts as shown in the figure 2:

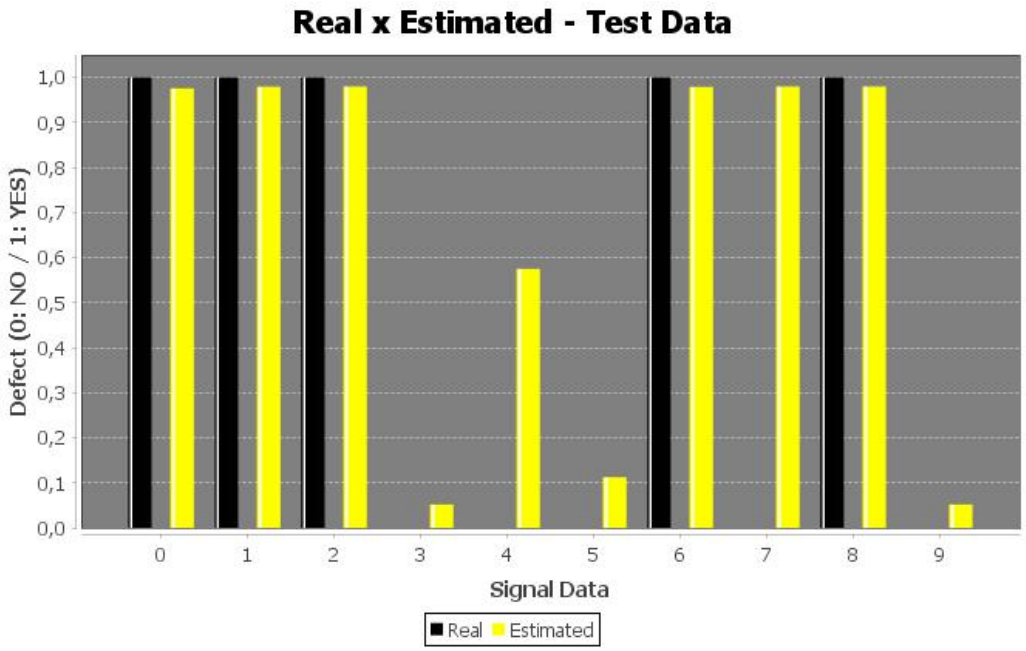

Figure 2 - The expected class along with the classification estimated by the neural network

Testing was performed as follows: Noise with an average value of 0 and a standard deviation from 0 to 0.2 in increments of 0.05 was added to the input vectors. For each noise level we calculated and constructed histograms the expected class along with the classification estimated by the neural network.

Conclusions. Classification tasks belong to one of the most frequently used types of supervised tasks in the fields of machine learning/data mining, and neural networks proved to be very appropriate for application to such problems. In article was presented with the concepts used for evaluating the classification tasks, such as sensitivity, specificity, and the confusion matrix.

In the hidden layer is better to take the number of neurons equal to half of the number of input values. The activation functions *hypertan* or *siglog* in the hidden layer and *sislog* in the output layer show the best results.

Training network on different sets of signals with noise allowed to teach her to work with distorted information, which is typical when conducted non-destructive testing in real conditions.

 ISSN 1562-9945

REFERENCES

1. Fábio M. Soares, Alan M.F. Souza, Neural Network Programming with Java, - Birmingham, 2016. - 244 p.

2. Haykin S. Neural Networks. A Comprehensive Foundation. Second edition. – New Jersey: Prentice Hall, 2008. -1103 p.

3. Herbert Schildt Java.The Complete Reference Ninth edition, 2014, 1372 p.

4. C. M. Bishop. Neural Networks for Pattern Recognition. Oxford University Press, 1995.

5. Хандецкий В.С. Спектральная идентификация сигналов в дефектоскопии композитов с использованием теории статистических испытаний / Хандецкий В.С., Герасимов В.В. //Вісник ДНУ: Фізика. Радіоелектроніка. – Дніпропетровськ: – 2003. № 10. – С. 128 – 132.

6. Матвеева Н.А. Моделирование нейросети для решения задачи классификации в дефектоскопии // Системні технології. Регіон. міжвуз. зб. наук. праць. - Дніпропетровськ: ДНВП «Системні технології», 2011. -Вип. 1(72). - С. 37-44.

REFERENCES

1. Fábio M. Soares, Alan M.F. Souza, Neural Network Programming with Java, - Birmingham, 2016. -244 p.

2. Haykin S. Neural Networks. A Comprehensive Foundation. Second edition. – New Jersey: Prentice Hall, 2008. -1103 p.

3. C.M. Bishop. Neural Networks for Pattern Recognition. Oxford University Press, 1995.

4. Herbert Schildt Java.The Complete Reference Ninth edition, 2014, 1372p.

5. Handetskiy V.S. Spektralnaya identifikatsiya signalov v defektoskopii kompozitov s ispolzovaniem teorii statisticheskih ispyitaniy / Handetskiy V.S., Gerasimov V.V. //VIsnik DNU: FIzika. RadIoelektronIka. – DnIpropetrovsk; 2003. №10 – S.128-132.

6. Matveeva N.A. Modelirovanie neyroseti dlya resheniya zadachi klassifikatsii v defektoskopii // SistemnI tehnologIYi. RegIon. mIzhvuz. zb. nauk. prats. - DnIpropetrovsk: «SistemnI tehnologIYi», 2011. -Vip. 1(72). - S. 37-44.