ISSN 1068-798X, Russian Engineering Research, 2022, Vol. 42, No. 3, pp. 278–281. © Allerton Press, Inc., 2022.
Russian Text © The Author(s), 2022, published in STIN, 2022, No. 1, pp. 25–28.

Neural Networks in Manufacturing

L. A. Simonova[a], *, E. I. Egorova[b], **, and A. I. Akhmadiev[b], ***

[a] *Kazan (Volga Region) Federal University, Kazan, Russia*

[b] *Tupolev Kazan National Research Technical University (Kazan Aviation Institute), Kazan, Russia*

**e-mail: lasimonova@mail.ru*

***e-mail: egrvelena@mail.ru*

****e-mail: blood_aid@mail.ru*

Received February 12, 2021; revised February 12, 2021; accepted February 12, 2021

Abstract—Attention focuses on minimizing the time to train a neural network so that it recognizes a specified set of a system's input parameters. In training the neural network, the error function must be minimized. This is important in expert assessment of solutions generated by a smart system for the design of manufacturing processes. In such a system, solutions are generated by the combined operation of numerous modules on the basis of logical rules. The system to be designed will generally be complex and may contain subsystems of different types that function according rules described by fuzzy logic and systems of precedents [1].

Keywords: neural networks, training, artificial intelligence, design solutions, fuzzy logic, cutting tools, manufacturing processes

DOI: 10.3103/S1068798X22030224

INTRODUCTION

In determining the optimal design solution, each component of the decision-making process is regarded as an individual system with its own characteristics and its own input and output parameters of the design process. The goal is to identify common features in the design process and to construct functional models of decision-making capabilities [2].

To assign the weighting factors in expert assessment of a smart system for designing industrial processes using artificial neural networks, genetic algorithms are employed. In conditions of indeterminacy, genetic algorithms offer a good chance of achieving the required results. By that means, neural networks may be adjusted and trained. The product to be designed was considered from the perspective of variable technological processes in [1]. To that end, a technology for displaying information regarding the design indices of the part was proposed in [3].

METHODS

We employ a genetic algorithm to train a neural network for the example in [4]. There are two tasks: (1) to select the optimal structure for the neural network; (2) to formulate an effective algorithm for training the network.

In optimizing the neural network, we want to minimize the volume of calculations, while maintaining the required accuracy of solution. The optimization parameters in the neural network are as follows.

1. The dimensions and structure of the network's input signal.

2. The neural synapses of the network. For the sake of simplicity, those that are least important are removed, or the required or optimal weight factors for the synapses are specified.

3. The number of neurons in each layer. On removal of a neuron from the network, those synapses in the next layer conveying its output signal are automatically removed.

4. The number of layers in the network.

The algorithm for effective training of the neural network involves adjusting the network so that the desired set of outputs corresponds to some set of inputs.

In training, the vectors of the training set are successively specified, with simultaneous adjustment of the weights by a predetermined procedure, until the error over the whole set is within acceptable limits.

The genetic algorithm is the best known of a class of evolutionary algorithms and is essentially a means of finding the global extremum of a function with numerous extrema. It consists of parallel analysis of a set of alternative solutions. The search is concentrated on the most promising candidates. This indicates the possibility of using genetic algorithms in any problem involving artificial intelligence, optimization, and decision making.

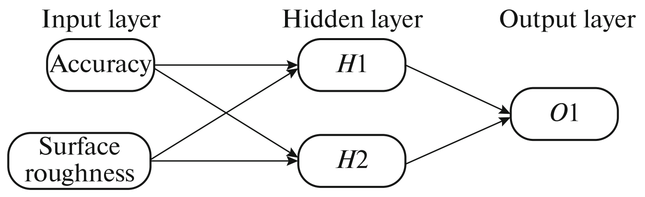

Fig. 1. Three-layer neural network.

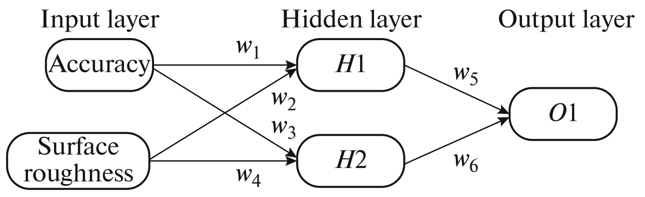

Fig. 2. Weighted neural network.

RESULTS AND DISCUSSION

For a structural element—specifically, a hole—we train a neural network of multilayer perceptron type, with the following parameters: a) the input parameters are the surface roughness and accuracy; b) there are three layers: the input layer *1*, intermediate layer *2*, and output layer *3* (Fig. 1) [1].

This element is described by a table of correspondences, with specific numerical values of the input parameters (Table 1).

After completing the calculations by the genetic algorithm for the structural element (the hole), we assign the identifier 1 to milling, and 0 to drilling.

To train the network, it is best to consider the mean square error (*MSE*) of the losses

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(o_{\text{true}} - o_{\text{pred}}\right)^2,$$

where n is the number of objects considered ($n = 3$ in the present case); and o are the predicted variables. In the present case, o_{true} is the true value (the correct answer); and o_{pred} is the predicted value (the network output).

We assume that our neural network always generates the output 0. In other words, for all the structural elements, the network proposes drilling (Table 2).

The mean square error of the loss due to the neural network output (Table 2) is $MSE = \frac{1}{4} \times (0 + 1 + 0 + 1) = 0.5$. A fragment of the program for the calculation is as follows:

```
import numpy as np
defmse_loss(o_true, o_pred):
# o_true and o_pred are numpy sets with the same length
return ((o_true - o_pred) ** 2).mean()
o_true = np.array([1, 1, 0, 1])
o_pred = np.array([0, 0, 0, 0])
print(mse_loss(y_true, y_pred)) # 0.75
```

We now modify the network prediction: we introduce weights by means of multivariate calculus. For the sake of simplicity, we consider only hole 1 from Table 2. Its characteristics are shown in Table 3.

In this case, the mean square error of the loss is

$$MSE = \frac{1}{1}\sum_{i=1}^{1}\left(o_{\text{true}} - o_{\text{pred}}\right)^2$$
$$= \left(o_{\text{true}} - o_{\text{pred}}\right)^2 = \left(1 - o_{\text{pred}}\right)^2.$$

The weights employed are shown in Fig. 2.

Writing the loss as a multivariate function

$$P\left(w_1, w_2, w_3, w_4, w_5, w_6\right),$$

we consider how the loss P changes if one of the weights is modified. To that end, we determine the partial derivative $\frac{\partial P}{\partial w_1}$

$$\frac{\partial P}{\partial w_1} = \frac{\partial P}{\partial o_{\text{pred}}}\frac{\partial o_{\text{pred}}}{\partial w_1};$$

$$\frac{\partial P}{\partial o_{\text{pred}}} = \frac{\partial\left(1 - o_{\text{pred}}\right)^2}{\partial o_{\text{pred}}} = -2\left(1 - o_{\text{pred}}\right).$$

Table 1

Element (n)	Accuracy	Surface roughness	Operation
Hole 1	8	6.3	Milling
Recess 1	6	3.2	Milling
Hole 2	14	12.5	Drilling
Recess 2	8	6.3	Milling

Table 2

Element (n)	o_{true}	o_{pred}	$\left(o_{\text{true}} - o_1\right)^2$
Hole 1	1	0	1
Recess 1	1	0	1
Hole 2	0	0	0
Recess 2	1	0	1

Table 3

Element (n)	Accuracy	Surface roughness	Operation
Hole 1	8	6.3	Milling

To determine $\frac{\partial o_{\text{pred}}}{\partial w_1}$, we need $H1$, $H2$, and $O1$ to correspond to the outputs of the neurons that they represent

$$o_{\text{pred}} = O1 = f(w_5H1 + w_6H2),$$

where f is the sigmoid activation function.

Since w_1 only affects $H1$, we may write

$$\frac{\partial o_{\text{pred}}}{\partial w_1} = \frac{\partial o_{\text{pred}}}{\partial H1}\frac{\partial H1}{\partial w_1};$$

$$\frac{\partial o_{\text{pred}}}{\partial H1} = w_5 f'(w_5H1 + w_6H2).$$

We determine the partial derivative $\frac{\partial H1}{\partial w_1}$ analogously

$$H1 = f(w_1X1 + w_2X2)$$

and

$$\frac{\partial H1}{\partial w_1} = X1f'(w_1X1 + w_2X2),$$

where $X1$ is the accuracy, and $X2$ is the surface roughness. In addition, $f'(X)$ is the derivative of the sigmoid function, which is appearing for the second time. We may write

$$f(X) = \frac{1}{1+e^{-x}}$$

and

$$f'(X) = \frac{e^{-x}}{(1+e^{-x})^2} = f(X)(1-f(X)).$$

We determine $\frac{\partial P}{\partial w_1}$ by inverse error propagation

$$\frac{\partial P}{\partial w_1} = \frac{\partial P}{\partial o_{\text{pred}}}\frac{\partial o_{\text{pred}}}{\partial H1}\frac{\partial H1}{\partial w_1}.$$

We now consider all these formulas for an example. The numerical values for the calculation are given in Table 4.

We specify unit weights. The neural network yields

$$H1 = f(w_1X1 + w_2X2) = f(8 + 6.3) = 0.9999;$$

$$H2 = f(w_3X1 + w_4X2) = f(8 + 6.3) = 0.9999;$$

$$O1 = f(w_5H1 + w_6H2) = f(8 + 6.3) = 0.8807.$$

Hence, we find that $o_{\text{pred}} = 0.8807$. This gives little guidance in choosing milling or drilling. Next we calculate $\frac{\partial P}{\partial w_1}$

$$\frac{\partial P}{\partial w_1} = \frac{\partial P}{\partial o_{\text{pred}}}\frac{\partial o_{\text{pred}}}{\partial H1}\frac{\partial H1}{\partial w_1};$$

$$\frac{\partial P}{\partial o_{\text{pred}}} = -2(1 - o_{\text{pred}}) = -2(1 - 0.8807) = -0.238;$$

$$\frac{\partial o_{\text{pred}}}{\partial H1} = w_5 f'(w_5H1 + w_6H2)$$
$$= 1f'(0.9999 + 0.9999) = f(1.9998)(1 - f(1.9998)$$
$$= 0.88070.1192 = 0.105;$$

$$\frac{\partial H1}{\partial w_1} = X1f'(w_1X1 + w_2X2) = 8f'(8 + 6.3)$$
$$= 8f(14.3)(1 - f(14.3)) = 4\times10^{-6};$$

$$\frac{\partial P}{\partial w_1} = -0.238\times0.105\times4\times10^{-6} = -0.123\times10^{-8}.$$

On that basis, we conclude that the computational error will increase if the weight is increased.

Now we have all the necessary tools for training the neural network. We consider optimization on the basis of stochastic gradient descent (SGD), which will indicate exactly how to change the weight so as to minimize the losses. We may write the equation: $w_1 \leftarrow w_1 - \eta\frac{\partial P}{\partial w_1}$, where η is a constant (the training estimate).

The training estimate determines the rate of training. All we need to do is to subtract $\frac{\partial P}{\partial w_1}$ from w_1. If $\frac{\partial P}{\partial w_1}$ is positive, w_1 will decrease, leading to decrease in P. If $\frac{\partial P}{\partial w_1}$ is negative, w_1 will increase, leading to increase in P. If we apply this technique to each weight in the neural network, the error will gradually decline, and the output parameters will improve.

The following algorithm may be used to train the neural network.

1. Select one point from the data set in Table 4, and apply stochastic gradient descent. Consider one point at a time.

2. Calculate all the partial derivatives of the losses, by weight. These may include $\frac{\partial P}{\partial w_1}$, $\frac{\partial P}{\partial w_2}$, and so on.

Table 4

Element (n)	Accuracy	Surface roughness	Operation
Hole 1	8	6.3	Milling

Table 5

Surface roughness	Accuracy							
	*IT*6	*IT*7	*IT*8	*IT*9	*IT*10	*IT*11	*IT*12	*IT*13
0.2	0.997	0.997	0.997	0.997	0.997	0.997	0.997	0.997
1.25	0.997	0.997	0.997	0.997	0.997	0.997	0.997	0.997
3.2	0.993	0.997	0.997	0.997	0.997	0.997	0.997	0.997
6.3	0.007	0.017	0.167	0.888	0.990	0.996	0.997	0.997
12.5	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005

3. Use the renewal equation to update each weight and displacement.

4. Return to step 1.

CONCLUSIONS

In a smart design system for technological processes, decisions are made on the basis of the algorithm for formulating and training a neural network with two input parameters: accuracy and surface roughness.

In training, the main goal is to minimize the error function, which may be written in the form [5]

$$E(w) = \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{m}\left(y_j^{(i)} - d_j^{(i)}\right)^2 \to \min,$$

where i is the number of the training sample; j is the number of the output neuron; p is the total number of training samples; m is the total number of output neurons; $y_j(i)$ is the signal of output neuron j for training sample i; and $d_j(i)$ is the expected value of training sample i for output neuron j.

The neural network is recalculated for a predetermined structure with weighting factors obtained by means of a genetic algorithm. In the present case, a genetic algorithm is most effective, since it yields some set of alternatives that are sufficiently close to the optimal value. These are weighted estimates for the technological process, which depend on the two chosen input parameters: accuracy and surface roughness. Table 5 summarizes the results given by the algorithm.

Thus, we have described a training method for a neural network used in a smart system intended for the design of manufacturing processes. The method permits the calculation of quantitative process parameters affecting the quality of the machined surfaces and the overall product cost.

REFERENCES

1. Simonova, L., Egorova, E., and Akhmadiev, A., Knowledge acquisition for engineering decisions based on functional relationships, *Int. J. Emerging Trends Eng. Res.*, 2020, vol. 91, pp. 2774–2778.
2. Simonova, L.A. and Egorova, E.I., Modular representation of the product in the knowledge base in the technological process formation, *Int. Sci. Conf.*, 2015, no. 258, pp. 6537–6540.
3. Egorova, E.I., Representation of information about part on the basis of its engineering features international, *J. Innovative Technol. Explor. Eng.*, 2019, vol. 8, no. 12, pp. 4084–4089.
4. Solomka, Yu.I., Application of genetic algorithms for training neural networks, *Materialy IV studencheskoi nauchno-prakticheskoi konferentsii "Mir molodezhi—molodezh' mira"* (Proc. 4th Student Sci.-Pract. Conf. "The World of Youth—the Youth of the World"), Vinnitsa: Vinnits. Inst. MAUP, 2004, ch. 1, pp. 85–90.
5. Haykin, S., *Neural Networks. A Comprehensive Foundation*, Upper Saddle River, N.J.: Prentice Hall, 1998.

Translated by B. Gilbert