The estimation with Artificial Neural Networks of some quality parameters for the surfaces processed by superfinishing

Daniel-Constantin Anghel^{1,*}, and *Alexandru Ene*²

¹University of Pitesti, Department of Manufacturing and Industrial Management, Tg. din Vale Str., no.1, Pitesti, Romania Romania

²University of Pitesti, Department of Electronics, Computers, Communications and Electrical Engineering, Tg. din Vale Str., no.1, Pitesti, Romania

Abstract. This paper presents a study on the quality parameters obtained by superfinishing. The quality is characterized by the roughness. They are dependent on the following process parameters: circular feed, the contact pressure between the piece and the tool, frequency of oscillation of the tool, the coverage degree between the tool and the piece and the basic time. Because the dependence between inputs and outputs is a nonlinear one, in this paper we used an artificial feed forward neural network (ANN). The ANN is trained with the backpropagation algorithm, using as training patterns data measured from the mechanical process. The ANN is used to estimate some parameters from future experiments of the mechanical process.

1 Introduction

One of the basic demands for the new products launched on the market is represented by the quality level. Because the clients are more and more exigent with the products, the manufacturers must give a distinct attention to quality. A part of general quality is represented by the quality of the processing surfaces.

As presented [1, 2], the analyse of the manufacturing parameters effect over the quality indexes shows that only a part of them has a significant effect to be considered and studied.

Processing by superfinishing is part of the final processing of surfaces with severe quality requirements. The authors conducted a research on the quality parameters for the surfaces processed by superfinishing.

The quality is characterized by the roughness (Ra and Rz) and they are dependent on the following process parameters: Vs, p, f, Ga and tb:

- Vs is the rotation speed of the piece (circular feed);
- p – the contact pressure between the piece and the tool;
- f – the frequency of oscillation of the tool;

* Corresponding author: daniel.anghel@upit.ro

- G_a – the coverage degree between the tool and the piece defined as the ratio between the length of the contact arc and the length of the circle in the cross section of the piece [1];
- t_{be} – the basic time actually scheduled.

The material of the part is an Fgn 700 – 2 iron (EN-GJS-700-2 cast iron, according to SR EN 1563:1999 standard).

For the study presented in this paper, a number of 32 experiments was chosen. These experiments are showed in the Table 1.

Table 1. The experiments of superfinishing.

Independent variables					Dependent variables	
					R_{aISO}	R_{zISO}
v_s [m/min]	p [bar]	f [cd/min]	G_a	t_{be} [s]	[μm]	[μm]
4.1	1.5	1250	0.2	118	0.1	0.52
12.86	1.5	1250	0.2	15	0.36	1.66
4.1	2.16	1250	0.2	15	0.22	1.31
12.86	2.16	1250	0.2	118	0.17	0.89
4.1	1.5	1800	0.2	15	0.29	1.47
12.86	1.5	1800	0.2	118	0.16	0.93
4.1	2.16	1800	0.2	118	0.19	0.71
12.86	2.16	1800	0.2	15	0.35	1.6
4.1	1.5	1250	0.4	15	0.23	0.95
12.86	1.5	1250	0.4	118	0.15	0.75
4.1	2.16	1250	0.4	118	0.08	0.53
12.86	2.16	1250	0.4	15	0.23	1.04
4.1	1.5	1800	0.4	118	0.17	0.99
12.86	1.5	1800	0.4	15	0.19	0.96
4.1	2.16	1800	0.4	15	0.20	1.00
12.86	2.16	1800	0.4	118	0.15	0.76
2.31	1.8	1500	0.28	42	0.17	0.87
22.76	1.8	1500	0.28	42	0.19	1.00
7.261	1.25	1500	0.28	42	0.12	0.60
7.261	2.59	1500	0.28	42	0.16	0.77
7.261	1.8	1041.6	0.28	42	0.15	0.78
7.261	1.8	2160	0.28	42	0.17	0.80
7.261	1.8	1500	0.14	42	0.25	1.16
7.261	1.8	1500	0.56	42	0.11	0.65
7.261	1.8	1500	0.28	5.36	0.27	1.42
7.261	1.8	1500	0.28	330	0.10	0.65
7.261	1.8	1500	0.28	42	0.15	0.75
7.261	1.8	1500	0.28	42	0.14	0.77
7.261	1.8	1500	0.28	42	0.15	0.79
7.261	1.8	1500	0.28	42	0.14	0.77
7.261	1.8	1500	0.28	42	0.14	0.71
7.261	1.8	1500	0.28	42	0.14	0.77

Because the dependence between inputs and outputs is a nonlinear one, in this paper we used an artificial feed forward neural network (ANN).

2 Feed Forward Neural Networks

Feed forward neural networks are intelligent devices that are able to recognize images, to approximate non-linear functions or to estimate different parameters. For instance, they are widely used to recognize hand written letters or digits. A classical successful application of neural networks, used in some post offices is to classify the received letters using the postal code number, and this postal code is hand written on the envelope [3]. They recognize images in the same manner as we, human beings, recognize them. We do not recognize a hand-written digit using a mathematical model of that digit. We recognize it, because, in the past, we have seen many examples of that digit. We learn through examples. The feed forward neural network learns to recognize or to classify an image, through examples, too. These examples are called training patterns.

A feed forward neural network consists of layers of interconnected artificial neurons. There is an input layer, one or more hidden layers and an output layer. The neurons in the same layer are not interconnected, but each neuron from a layer is totally connected with all the neurons from the next layer. This organization of artificial neurons in layers is biologically inspired (we have a sensorial layer of neurons, layers of processing neurons in cortex and a layer of motor neurons). The strength of a connection between two artificial neurons, is simulated associating to that connection, a real value, positive or negative, called weight (in the biological model, we have excitatory connections between neurons, that correspond to the positive weights, and we have inhibitory connections that correspond to negative weights).

An input layer could have binary inputs (as is typically in pattern recognition) and/or analogue inputs.

In Figure 1 is presented a 3-2-2 feed forward neural network (3 neurons in the input layer, 2 neurons in the middle layer i.e. in the single hidden layer, and 2 neurons in the output layer).

Fig. 1. A 3-2-2 feed forward neural network.

The neurons from the input layer have an identity transfer function ($\text{output}=\text{input}$), and they have the role to distribute the current input to all the neurons from the next layer.

The other neurons from the other layers have a sigmoidal transfer function, in which: $\text{output}=\frac{1}{1+\exp(-\text{activation})}$.

The activation for an artificial neuron is defined as the weighted sum of all its inputs. This is a nonlinear transfer function and this nonlinearity makes the ANN capable to approximate nonlinear functions. And this is the second major application of feed forward neural

networks. And for this reason, we used in this paper a feed forward neural network to model the nonlinear relationship between some mechanical parameters and the two quality parameters.

In [4] is presented a feed forward neural network that is able to learn a nonlinear process described by a single input and a single output. In this paper, we developed a neural network that is able to learn a nonlinear process, but it has more inputs.

A feed forward neural network can be implemented in many different ways:

- A hardware implementation, using electronic VLSI neural chips. This offers the greatest speed for processing, but is the least flexible.
- by software simulation, using parallel computers
- by software simulation using common programming languages, on ordinary computers This is the most flexible approach, but is not fast.
- software simulation using special neural software (for instance Matlab with its Neural ToolBox)

In order to use a neural network for a specific task (either pattern recognition or nonlinear function approximation), firstly, the architecture of the network has to be determined: i.e. the number of inputs, the number of hidden layers, the number of neurons from each hidden layer, and the number of neurons from the output layer.

The number of neurons from the input layer is set by the application. In our application we have five mechanical parameters, so we will use five neurons in the first layer.

The number of neurons in the output layer is also set by the problem to be solved.

The number of hidden layers and the number of neurons in each hidden layer is experimentally chosen. Typically, an ANN that is used for pattern recognition has one hidden layer, but an ANN that is used for nonlinear function approximation has two hidden layers. So, in our application we used two hidden layers, each having five artificial neurons.

After this stage (determining the architecture) we have to train the network. Through the training, all the weights from the network are computed. We can say that the “intelligence” of the network stays in its weights.

When the network is software simulated, the values of the weights are saved in a computer file.

After the network is trained, it can be used to solve the real problem for which it was trained. In the case of a pattern recognition problem, the image to be recognized is placed on networks’ inputs (generally, each pixel from the image is connected to an input neuron) , and the network computes the outputs of all neurons from the hidden layer and after this, it calculates the outputs of all neurons from the output layer. The output neuron which has the biggest output corresponds to the class from which the input image belongs. In the case of our approximation problem, after the input data are placed on the inputs of the network, the network calculates the outputs of all neurons from the two hidden layers, and then the outputs of neurons from the output layer.

3 The software application

Firstly, we used for our application a 5 – 5 – 5 -2 feed forward neural network. But, through simulations, we decided that is better to use a 5 – 5 – 5 – 1 neural network, that is firstly trained to estimate the Ra parameter (the set of weights for Ra is calculated), and then, the same architecture 5 – 5 – 5 – 1 is trained to estimate the Rz parameter (a new set of weights is calculated). The network that we used, learns the training patterns faster than the network 5-5-5-2. The network is simulated with Java programming language.

An example of a training pattern for Ra:

Inputs: 4.1, 1.5, 1250, 0.2, 118

Output: 0.1

An example of a training pattern for Rz:

Inputs: 4.1, 1.5, 1250, 0.2, 118

Output: 0.52

These data, in order to be successfully processed by the neural network, had to be scaled, because all sigmoidal neurons of the network have an output in the interval (0, 1). So, for each from the five input parameters and also for the two output parameters, we calculate the maximum (max) and the minimum (min) from all the experimental training patterns. In order to scale a value x, from [min, max] interval to a value y in an [A, B] interval, we used a linear scaling, given by the formula:

$$y=(x*(B-A)+A*\max-B*\min)/(\max-\min) \quad (1)$$

In this application, good results had been obtained with A=0.1 and B=0.9.

The feed forward neural network is trained using the back-propagation algorithm. This algorithm iteratively calculates all the weights from the network, based on available training patterns.

We say that the network converges when it learns with a specified error, all the training patterns. In this application the learning error was 1%. So, in our case, the network converges when total learning error <0.01

The back-propagation algorithm starts initializing all the weights with small random values, real numbers positive and negative.

The back-propagation algorithm continues in the following manner:

```

WHILE the network did not converge REPEAT
  FOR all training patterns REPEAT
    begin
      calculate the outputs of all neurons (using the
        current pattern)
      compute the error of learning current pattern
      modify all the weights using this error (the delta
        rule of the backpropagation algorithm)
    end
    calculate the total error of learning all the patterns
    (as the sum of individual learning error for each
    pattern)
  
```

A training epoch is done when all the training patterns had been presented on the inputs of the network, its weights were modified according to the delta rule and the total error had been computed.

After the training, the network is tested with data that it did not learn (data that are different from the training patterns), in order to see the ability of network to generalise. For the test, we take a set of values for the 5 input parameters: Vs, p, f, Ga and tb, scale them, and place them on the 5 inputs of the network, and the program computes the output of the network (using the weights calculated by the back-propagation algorithm). In order to compare the value obtained at the output of the network (y) with the real value, the output of the network must be re-scaled, using the formula:

$$y_r=(y*(\max-\min)-A*\max+B*\min)/(B-A) \quad (2)$$

4 The experimental results

Our 5 – 5 – 5 – 1 feed forward neural network converges in tens of thousands of epochs, for a 1% learning error. (of course this number is variable, depending on the random values that

had been used for the initialization of the weights). The speed of convergence for the network depends also of the scaling interval [A, B].

We tested the network with values that it did not learned. For example, for the following set of input values: 12.86, 1.5, 1250, 0.2, 15, the value measured for Rz was: 1.66. The value computed by the network was 1.71. The network converged in about 60,000 epochs, as illustrated by the following execution of the program:

THE TRAINING PHASE:

Epoch 1: total learning error=3.398574329350182

Epoch 2: total learning error=3.11323717907116

.....

Epoch 59901: total learning error =0.010000067948527896

Epoch 59902: total learning error=0.00999888421416941

THE NETWORK COVERGED !

THE TESTING PHASE:

Input values: 4.1, 1.5, 1250, 0.2, 118

Correct value: 0.52, neural network computed value 0.5202859669811853

Input values: 12.86, 1.5, 1250, 0.2, 15

Correct value: 1.66, neural network computed value: 1.7115214289464546

The first set of input values, used in the testing phase was a pattern used for the training of the network. That is why the correct value and the computed value were quite the same. But the second set of input values were not used in the training, and still, the error is acceptable.

5 Conclusions

In this paper, we used a feed forward neural network, to approximate the parameter of quality of the surface, Ra and Rz. The networks have only analogue inputs.

The necessary data for training and for testing the neural network is half of necessary data than using the method of process functions (classical method), which is a considerable advantage in financial terms, in terms of availability of machine tools etc.

The approximation error could be improved if we had more available experimental data, in order to use much more training patterns.

References

1. Il. Banu, *PHD Thesis*, ("Transilvania" Univ. of Braşov, 1999)
2. Il. Banu, V. Nicolae, Int. Conf. TMCR 2006, Iasi, May, 25th – 27th (2006)
3. A. Ene, C. Stirbu, *Neural networks* (in Romanian), (Ed. Univ. Pitesti, 2008).
4. A. Ene, C. Stirbu, ECAI 2016, Ploiesti, DOI: 10.1109/ECAI.2016.7861197 (2016).