Int. J. Econ. Environ. Geol. Vol. 13 (3) 01-10, 2022

Journal home page: www.econ-environ-geol.org

Open Access

ISSN: 2223-957X

Estimation of Calorific Value of Lignite Field in Kütahya-Gürağaç (Turkey) by means of Artificial Neural Network

Sedat Toraman[1], Cem Şensöğüt[2*]

[1] General Directorate of Turkish Coal Enterprises, Ankara, Turkey
[2*] Kütahya Dumlupinar University, Mining Engineering Dept., Kütahya, Turkey

*Email: cem.sensogut@dpu.edu.tr

Abstract: Artificial neural networks are generally information processing systems that mimic the working principles of the human brain or central nervous system. Artificial neural networks are a method that gives successful results in solving many daily life problems such as classification, modeling and prediction. Artificial neural networks accomplish this by adjusting the connection weights between neurons. It can solve prediction and classification problems with back propagation algorithm, which is widely used in artificial neural networks with multilayer perceptron. In this study, unknown calorific values were tried to be estimated by using the analysis values (depth, ash, moisture, sulfur, calorific value) of the drillings realized in the Kütahya -Gürağaç lignite field. An artificial neural network was created for this purpose. First, 8 neurons were used in the hidden layer of the network, and 10 neurons were used secondarily. In the artificial neural network, the learning function is sigma, the learning rate is 95%, and the network is trained using Levenberg-Marquardt as the training algorithm. The network with 10 neurons converged at the desired margin of error (1e-07) and was completed after 271 iterations. The relationship between actual calorie values and predicted calorie values with network training reached a high ratio of R^2=0.97. After the training of the network is completed, the network is simulated for the estimation of seams with unknown caloric values. As a result, caloric values were determined with an average of 97% confidence interval for the unknown coal seams of the field.

Keywords: Artificial neural networks, calorific value, coal seam.

Introduction

Artificial neural networks (ANN) are information processing systems that generally imitate the working principles of the human brain or central nervous system (Freeman and Skapura, 1991). Studies on this subject first started with the modeling of neurons, which are the biological units that make up the brain, and their application in computer systems. Neurons are interconnected by connections, and each connection has a numerical weight that expresses the strength, or in other words, the importance of its input. Weights are the main tool of long-term memory in ANNs. A neural network learns by repeatedly adjusting these weights (Negnevitsky, 2005). The generalization ability of the artificial neural network is directly related to the correct selection of the topology of the network. The optimal architecture for the network should be large enough to learn about the problem and small enough to generalize. A network that is smaller than the most suitable architecture cannot learn the problem well, on the other hand, a larger network over-learns the training data, which causes it to memorize and therefore has poor generalization ability. There are basically two greedy approaches to determining the structure of the network: growing/constructive and pruning/destructive. If the structure of the network is chosen small and grows during the learning process, a growing/constructive approach is followed; on the contrary, a pruning/destructive approach is followed if it is chosen large and shrinks during the learning process (Aran et al., 2009).

Results and Discussion

Artificial Neural Networks

Artificial neural networks are generally divided into two as single-layer perceptron and multi-layer perceptron.

Single layer perceptron model

Single-layer artificial neural networks are used to solve linear problems and consist of only input and output layers. Layers may have one or more neurons. A simple single-layer perceptron model is shown in Figure 1.

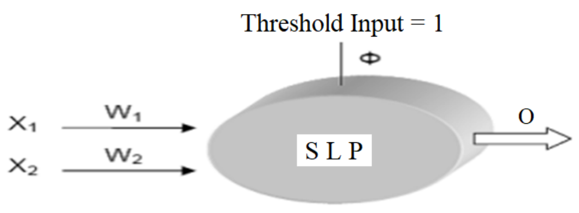


Fig. 1 Single layer perceptron model.

The threshold input prevents the values of the neuron elements and the output of the network from being 0 in such networks. Its value is always 1. The output of the network is obtained by summing the weighted input values with the threshold value as shown in Equation 1 (Arı and Berberler, 2017).

$$N_o = f\left(\sum_{i=1}^{n} w_i x_i + \Phi\right) \quad (1)$$

In Equation 1, xi, i = 1,2, …, n are the inputs of the network, w_i, i = 1,2, …, n are the corresponding weight values, and ϕ the threshold value. In a single-layer perceptron, the output function is linear. Thus, the examples shown to the network are shared between the two classes by the threshold function, and the line that separates the two classes is tried to be found. The output of the network takes a value of 1 or −1. The threshold function is shown in Equation 2.

$$f(g) = \begin{Bmatrix} 1 \; N_o > 0 \\ -1 \; N_o \leq 0 \end{Bmatrix} \qquad (2)$$

The class separator line is defined as in Equation 3.

$$w_1x_1 + w_2x_2 + \Phi = 0 \qquad (3)$$

From here;

$$x_1 = -\frac{w_2}{w_1}x_2 - \frac{\Phi}{w_1} \qquad (4)$$

$$x_2 = -\frac{w_1}{w_2}x_1 - \frac{\Phi}{w_2} \qquad (5)$$

is obtained as by using Equations 4 and 5, the class separator line, whose geometric representation is given in Figure 2, can be drawn.

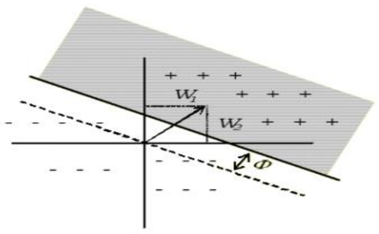


Fig. 2 Geometric representation of the class separator line.

Weight values are changed with the formula in Equation 6 at each iteration to determine the class separator line to best separate both groups (Arı and Berberler, 2017).

$$w_i(t+1) = w_i(t) + \Delta w_i(t) \qquad (6)$$

The threshold value is also updated with the formula in Equation 7 at each iteration to shift the class separator line between classes.

$$\Phi(t+1) = \Phi(t) + \Delta\Phi(t) \qquad (7)$$

There are two main models for single-layer sensors.

- ✓ Perceptron Model
- ✓ Adaline/Madaline Model (Öztemel, 2012).

a) Perceptron (Simple Sensor) Model

The perceptron model, developed by psychologist Frank Rosenblatt in 1958 to "simulate some of the basic properties of intelligent systems", is based on the principle that a nerve cell produces an output by considering more than one input. The output of the network is obtained by comparing the weighted sum of the input values with a threshold value. If the total is greater than or equal to the threshold, the output value is 1, and 0 if it is less. Rosenblatt developed a learning rule for sensor training that solves pattern recognition problems (Rosenblatt, 1958). He proved that this rule will always converge to the correct weights if there are weights that solve the problem. Marvin Minsky and Seymour Papert have publicly demonstrated that sensors can be used in very limited areas and that there are too many problem classes that the detector cannot solve, in their book "Perceptrons", as a result of their deep mathematical investigations on sensors (Minsky and Papert, 1969). An example of problems that sensors cannot solve is the XOR problem. This limitation of the sensors was eliminated with the development of the multilayer perceptron model in the 1980s.

b) Adaline Model

Bernard Widrow started working on neural networks in the late 1950s (Widrow, 1959). In 1960, Widrow and his graduate student Marcian Hoff developed a method called the Least Mean Square algorithm with the ADALINE network. This neuron model, with the clear name 'ADAptive LINEar NEuron' or 'ADAptive LINEar Element', does not differ much from the perceptron structurally. However, ADALINE considers the linear function while using the threshold function as the sensor activation function. In both models, there can be solutions only for linearly separable problems. The Least Squares algorithm, also called the Widrow-Hoff rule, is more powerful than the perceptron learning method. Even if the perceptron learning rule guarantees convergence to a solution, it can be noise sensitive due to the proximity of the training patterns to the borderline. The least squares algorithm tries to keep the training patterns as far from the boundary line as possible, as it minimizes the mean squared error. Widrow and Hoff also developed the MADALINE neural network model, which includes multiple adaptive elements (Widrow and Hoff, 1960).

Multi-layer perceptron model

The failure of single-layer perceptions to solve nonlinear problems has led to the development of multi-layer perceptron (MLP). These sensors consist of an input layer, one or more hidden (intermediate) layers, and an output layer. They also have transitions between layers called forward and backward propagation. In the forward propagation step, the output of the network and the error value are calculated. During back propagation, the interlayer link weight values are updated to minimize the calculated error value. The general structure of these sensors is shown in Figure 3. The back-propagation learning algorithm, which is a generalization of the least square's algorithm in the linear perceptron, is used in the MLP model.

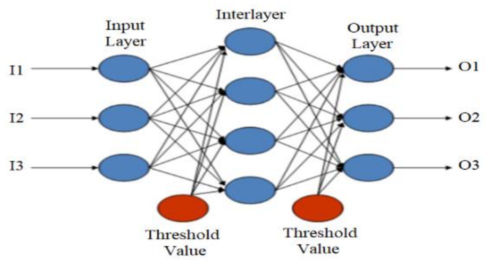


Fig. 3 The general structure of MLP.

Back Propagation Algorithm

The back-propagation algorithm consists of the stages where the output of the network is determined, the feed forward and the weights are updated by back propagation to reduce the gradient of the error. In the feed forward stage, the inputs of the training set are fed to the input layer of the network. The input layer contains neurons that accept these inputs. For this reason, the number of neurons in the input layer must be equal to the number of input values in the data set. The neurons in the input layer pass the input values directly to the hidden layer. Each neuron in the hidden layer calculates the total value by adding the threshold value to the weighted input values, and transmits them to the output layer by blending them with an activation function. The weights between the layers are usually randomly chosen at the beginning. After the net input of each neuron in the output layer is calculated by adding the threshold value to the weighted input values, this value is again processed with the activation function to determine the output values.

The error value is found by comparing the output values of the network with the expected output values. Therefore, the number of neurons to be found in the output layer must match the number of outputs in the data set. After the $\boldsymbol{n}$th training data for the $\boldsymbol{j}$th output cell, the error is defined as follows, with $\boldsymbol{d_j(n)}$ being the expected value;

$$e_j(n) = d_j(n) - y_j(n) \tag{8}$$

The total error in the output layer is expressed by Equation 9.

$$E\,(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n) \tag{9}$$

The set C contains all the neurons in the output layer. Here, $E(n)$ is tried to be minimized with an approach similar to the delta rule. The sum of the inputs to the output layer cell is expressed by Equation 10.

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n)x_i(n) \tag{10}$$

X = (x_1…x_n), j. m indicates the input value applied to the neuron, w_j indicates the weight of the x_i input, and f the activation function. w_{j0} denotes the deviation element so that x_0 = +1. The result produced by the output cells of the network is calculated by the formula in Equation 11.

$$y_i(n) = f(v_j(n)) \tag{11}$$

The gradient of the network can be found by differentiating the error function according to the weights. According to the chain rule, the gradient can be expressed as:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)}\frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{12}$$

If individual derivatives are taken,

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n)f(v_j(n))x_i(n) \tag{13}$$

The weight correction amount is applied according to the delta rule $\boldsymbol{\Delta w_{ji}(n)}$.

$$\Delta w_{ji}(n) = -\eta\frac{\partial E(n)}{\partial w_{ji}(n)} \tag{14}$$

$\boldsymbol{\eta}$ is the learning rate. The − sign in Equation 14 represents the steep descent in the weight space. Thus, the weight correction amount for the back-propagation algorithm is expressed as in Equation 15.

$$\Delta w_{ji}(n) = \eta\delta_j(n)x_i(n) \tag{15}$$

The local gradient $\boldsymbol{\delta_j(n)}$ is defined as given in Equation 16 (Arı and Berberler, 2017).

$$\delta_j(n) = e_j(n)f'(v_j(n)) \tag{16}$$

For any j neurons in the hidden layer, the desired output value is not specified, as are the neurons in the output layer. For this reason, the error value of a hidden j neuron will be affected by the error value of all neurons directly connected to that neuron. For any neuron j in the hidden layer, the local gradient $\delta j\ (n)$ is defined as in Equation 17 (Arı and Berberler, 2017):

$$\delta_j(n) = f'\big(v_j(n)\big)\sum_{j=0}^{1}\delta_j(n)W_{ji}(n) \tag{17}$$

By adding the momentum term α to the weight update equation of the back-propagation algorithm by Rumelhart et al. (1986), the probability of the mesh being stuck at the local minimum is reduced. After adding the momentum term, the weight update equation became as seen in Equations 18 and 19 (Arı and Berberler, 2017):

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \tag{18}$$

$$\Delta w_{ji}(n) = \eta\delta_j(n)x_i(n) + \alpha\Delta w_{ji}(n) \tag{19}$$

The learning methods used to include the training set in the calculation in the back-propagation algorithm are divided into two groups as single (online) and batch training methods. Updating the weights in single training is provided by back propagation of the error that occurs when each sample in the training dataset is applied to the network. In collective training, it is possible to update the weights by back propagation of the average error obtained after the entire training data set is applied to the network. While collective training can be parallelized, individual training cannot be parallelized (Haykin, 2009).

Heuristic approaches are used to reduce training times. Heuristic approach techniques, which are one of the few techniques to speed up convergence and improve the performance of the network in the back-propagation algorithm, are made using the momentum coefficient. The momentum coefficient is a factor that helps the ANN recover faster. It is basically based on the principle of adding a portion of the previous exchange to the traded exchange. Momentum coefficient not only allows the network to exceed local gradients, but also helps to reduce the error (Bayındır and Sesveren, 2008).

The learning rate (**η**) is a constant that controls and is proportional to the speed and accuracy of a learning procedure. The learning rate is used to change the weights of the ANN. If the learning rate is chosen too large, wide jumps occur in the error level, narrow areas where learning will take place can be skipped. Also, movements across the fault surface become very uncontrolled. If it is selected too small, the learning time may take a lot of time. Experience shows that the learning rate chosen in the range of $0.01 \leq \eta \leq 0.9$ gives good results. A large learning rate may lead to good results initially, but may lead to incorrect results later on. Using a smaller learning rate is more time consuming, but the result is clearer. Thus, in the learning process, the learning rate should be chosen large at the beginning and reduced over time (every iteration or every few iterations) (Kriesel, 2007). The reduction of the learning rate over time is called decay.

Estimated Calorific Values of Kütahya-Gürağaç Lignite Field

While the ash, moisture and calorie analysis results on the cores obtained from the drilling works were evaluated in the field coal seam modeling, it was determined that there was no calorific value in some analysis results. Although calorific values were determined by considering different solution methods, estimations were tried to be made with a maximum of 85% R^2 values. It was decided to detect unknown calorific values by using artificial neural networks in order to determine higher R^2 values and values closer to real calorific values.

By using the analysis results of ash, moisture and calorific values from 587 samples obtained from the field in concern, the artificial neural network was trained and as a result of the training, the calorific values were estimated as a result of 1401 analysis with only depth, ash and moisture values, but no calorific values.

Depth, ash and humidity values used as input variables before the mesh was formed were normalized between 0 and 1 for all values given in Table 1. Again, the normalization of the target variable, the calorific values, was also carried out. Normalized input and target variable matrices are transposed. Input and target matrices were formed as [3x587] and [1x587], respectively.

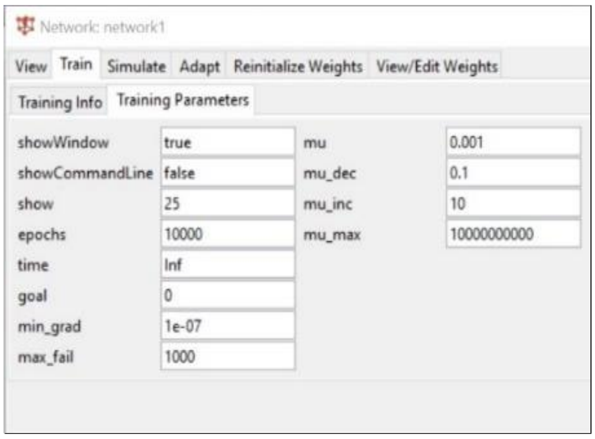
Normalization was performed in the estimation matrix, which has 1401 data whose calorific values are unknown, and then the matrix of [3x1401] was obtained by taking the transpose. As a result of all these processes, input data of [3x587], target data of [1x587] and prediction data of [3x1401] were prepared.

Table 1. Values used in Normalization of Data.

	Depth (m)	Humidity (%)	Ash (%)	Real Cal. Value (Kcal/kg)
Minimum	35	7,7	2,4	506
Maximum	185,26	34,5	80,29	5890
Max.-Min.	150,26	26,8	77,89	5384
1/(Max.-Min.)	0,006655131	0,037313433	0,012838619	0,000185736

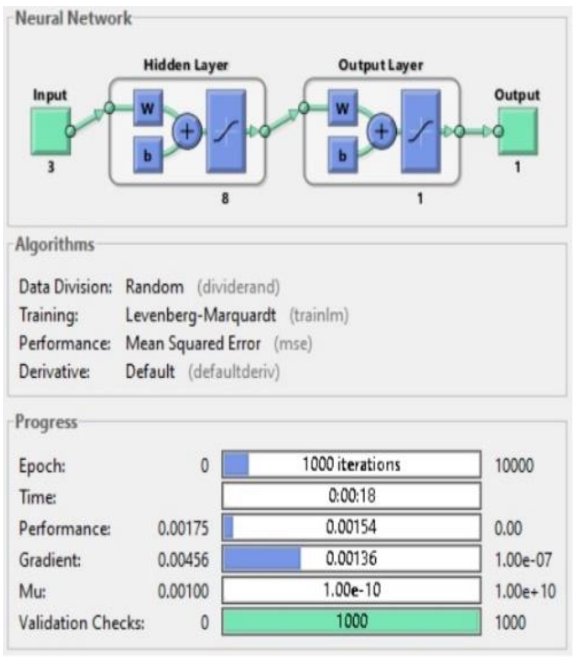
These prepared data sets were fed to the MATLAB program to be used in the neural network to be created. Training was carried out on MATLAB using 8 neurons in the hidden layer and Levenberg-Marquardt algorithm as a trainer. 70% of the input and target data with 587 data were used for training, 15% for validation and 15% for testing.

Network training was performed with the training parameters shown in Figure 4 and as a result of the training (Figure 5), R=0.982 for training, R= 0.985 for validation, and R=0.991 for testing were obtained (Figure 6). However, since the regression did not converge at the desired value (1e-07), the network was re-run by changing the number of neurons.



showWindow	true	mu	0.001
showCommandLine	false	mu_dec	0.1
show	25	mu_inc	10
epochs	10000	mu_max	10000000000
time	Inf		
goal	0		
min_grad	1e-07		
max_fail	1000		

Figure 4. ANN Training Parameters



Algorithms

Data Division:	Random	(dividerand)
Training:	Levenberg-Marquardt	(trainlm)
Performance:	Mean Squared Error	(mse)
Derivative:	Default	(defaultderiv)

Progress

Epoch:	0	1000 iterations	10000
Time:		0:00:18	
Performance:	0.00175	0.00154	0.00
Gradient:	0.00456	0.00136	1.00e-07
Mu:	0.00100	1.00e-10	1.00e+10
Validation Checks:	0	1000	1000

Fig. 5 ANN Network Training Simulation with 8 Neurons.

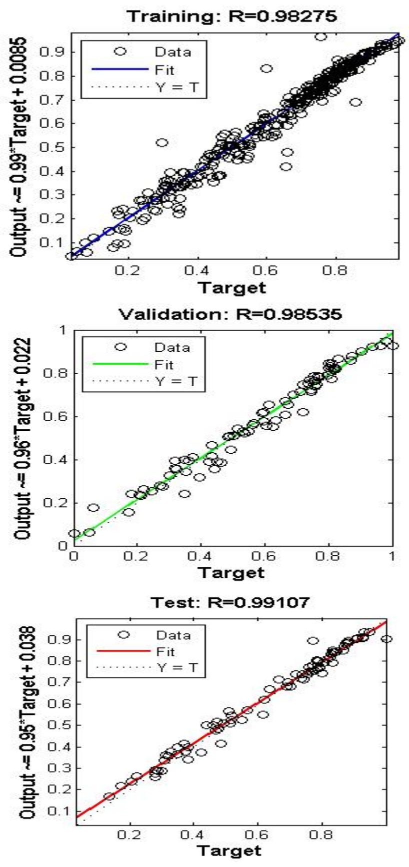
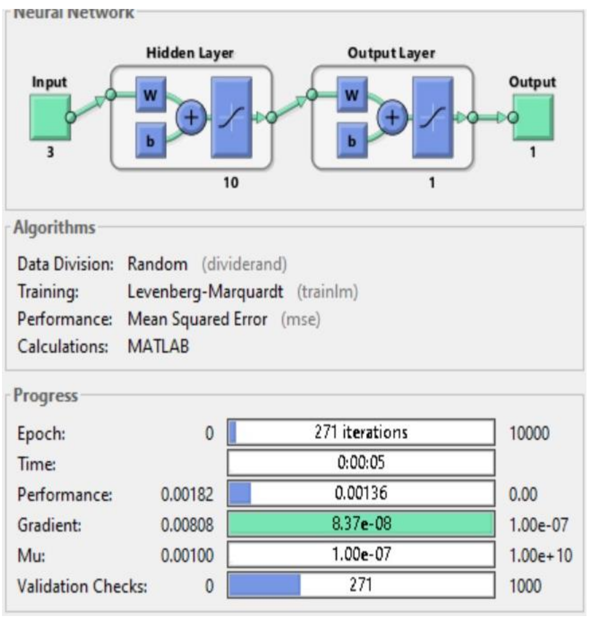


Fig. 6 Neuron Network Training Outputs.

The network training study was carried out using 10 neurons in the hidden layer with the same training parameters (Figure 7), and the artificial neural network with 10 neurons converged at the determined value (1e-07), and the training was completed as a result of 271 iterations. As a result of the training, the relationship between training, validation and test data was formed as in Figure 8.



Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MATLAB

Progress

Epoch:	0	271 iterations	10000
Time:		0:00:05	
Performance:	0.00182	0.00136	0.00
Gradient:	0.00808	8.37e-08	1.00e-07
Mu:	0.00100	1.00e-07	1.00e+10
Validation Checks:	0	271	1000

Fig. 7 ANN Network Training Simulation with 10 Neurons.

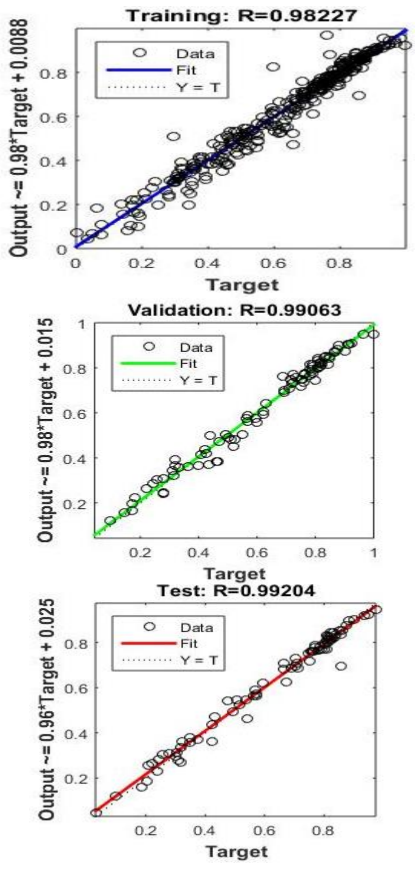


Fig. 8 Neuron Network Training Outputs.

In order to compare the calorific outputs of the neural network with 10 neurons with the real calorific outputs, inverse normalization was applied to the data output from the system. The actual calorific values of 587 analysis results and the calorific values obtained were overlapped (Figure 9). When the registration graph is examined, it is seen that the system makes estimations very close to the actual calorific values. It was determined that there is a very strong relationship such as R^2=0.97 between the Real Calorific values and the Estimated Calorific values.

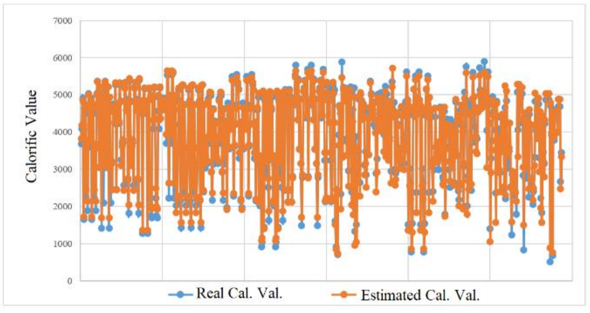


Fig. 9 Comparing Actual Calorific Values with Estimated Calorific Values.

After this stage, simulation was carried out for the trained network to produce calorific value in response to the analysis results whose calorific values are unknown. The calorific values produced by the network as a result of the simulation were added to the estimation data by inverse normalization. The table created for some calorific values was realized as in Table 2. All necessary data set for Kütahya-Gürağaç lignite field modeling is ready with the estimated calorific values with 97% probability.

Table 2. Calorific Values Estimation of Some Drilling Points with Unknown Calorific Values after Simulation.

No of Drill	Depth (m)	Moisture (m)	Ash (%)	Estimated Cal. Val. (Kcal/kg)
365	22,75	35,00	26,28	4848
408	55,00	18,50	47,24	2783
415	14,40	20,00	24,48	3844
491	37,80	29,00	16,28	5266
631	93,40	4,63	51,30	2309
758	13,50	8,00	15,80	5467
774	45,65	22,40	38,70	3293
803	13,50	22,00	39,20	3524
806	45,82	35,00	24,74	5096
832	32,60	15,00	24,96	4639
848	60,05	33,26	37,71	4140
1519	69,30	16,30	69,64	1240
2340	19,00	11,00	85,30	748
2405	10,65	16,30	56,32	1800
2578	36,50	15,00	57,38	2076
3085	81,25	17,60	29,80	4396

Conclusion

Coal field sample analysis results can be predicted quite successfully with artificial neural networks. With the artificial neural network created for the Kütahya-Gürağaç lignite field, the unknown calorie values were successfully estimated at a rate of 97% using known depth, ash and humidity values.

References

Aran, O., Yıldız, O. T., Alpaydın, E. (2009). An incremental framework based on cross-validation for estimating the architecture of a multilayer perceptron. *Int. J. Pattern Recognit.. Artif.. Intell.,* **23** (2), 159–190.

Arı A., Berberler M. E. (2017). Interface design for the solution of prediction and classification problems with ANN. *Acta Infologica*, **1** (2), 55-73, (in Turkish).

Bayındır, R., Sesveren, Ö. (2008). A visual interface design for ANN based systems. *J. of Engineering Sciences,* **14** (1), 101–109, (in Turkish).

Freeman, J. A., Skapura, D. M. (1991). Neural networks algorithms, applications and programming techniques. New York, USA: Addison-Wesley Publishing Company.

Haykin, S. (2009). Neural networks and learning machines. **3**, Pearson. Upper Saddle River, N.J., USA

Kriesel, D. (2007). A brief introduction to neural networks, Zeta, 2nd Ed. available at http://www.dkriesel.com.

Minsky, M., Papert, S. (1968). Perceptions. M.I.T. Press, USA.

Negnevitsky, M. (2005). Artificial intelligence: A guide to intelligent systems. Pearson Education.

Öztemel, E. (2012). artificial neural networks. Papatya Press & Education, Istanbul, (in Turkish).

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.,* **65** (6), 386–408.

Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature,* **323**, 533-536.

Widrow, B. (1959). Adaptive sampled-data systems—a statistical theory of adaptation. Paper presented at the IRE WESCON Convention Record.

Widrow, B., Hoff, M. E. (1960). Adaptive switching circuits. Paper presented at the WESCON Convention Record Part IV.