IOP Conf. Series: Materials Science and Engineering **427** (2018) 012012 doi:10.1088/1757-899X/427/1/012012

Modelling of oil agglomeration of dolomite by the use of an artificial neural network optimized using Optimal Brain Damage algorithm

M Kamiński[1] and A Bastrzyk[2]

[1]Faculty of Electrical Engineering, Wroclaw University of Science and Technology, Janiszewskiego 8, 50-377 Wroclaw, Poland

[2] Faculty of Chemistry, Wroclaw University of Science and Technology, Norwida 4/6, 50-373 Wroclaw, Poland

anna.bastrzyk@pwr.edu.pl

Abstract. This paper investigates application of Artificial Neural Network (ANN) for dolomite oil agglomeration modelling including parameters such as surfactants concentration, oil dosage, time, pH and mixing intensity. The main algorithm implemented for weights calculation was the Levenberg-Marquardt (LM) method. Common problem during process design of neural models is suitable selection of structure complexity. It is known that several connection can influence on final results after off-line training. For improvement of this stage of preparation of the net, pruning method was implemented. Analysed algorithm was based on the main theory of Optimal Brain Damage (OBD) technique. Results present high quality of process recovery prediction. Achieved outcome also shows that reduction of the applied of the neural network can lead to higher precision of calculation.

1. Introduction

Neural network is a flexible model, that can be adapted for realization of specific task in training process. It is useful attribute in a wide range of engineering applications, successful implementation of those models are noted in following fields: control (for instance of electrical drives) [1]-[3], state variables estimation [4], faults detection [5], signal prediction [6], data and images processing [7], etc. Based on papers published in journals and conference proceedings, three main – the most popular – group of structures can be mentioned: Multi-Layer Perceptron (MLP), Radial Basis Function Neural Networks (RBFNN) and Recurrent Neural Networks (RNN) [8], [9]. Characteristic feature of the first of them is one direction of input signals propagation inside of neural network. Moreover, mapping of data space is realized as global approximation using combination of several function of activation functions [10]. Neural networks with radial functions applied in hidden layer, can analyse data grouping following values in clusters (locally) and then aggregation is realized as weighted sum of signals. The division of input elements, realized with those models, is often used in the classification of data. Last significant structures of networks are RNN, those networks contain additional memory elements implemented in inner feedback connections. It is important advantage for processing of dynamic, time changeable variables.

Presently, growing number of neural network application is also being observed in chemistry. Multi-Layer Perceptron neural network can be applied for flotation process of coal fines modelling

[11]. Implementations of RBFNN are also noticed for similar task [12], moreover authors present comparison between different method of modelling used for this purpose. Other example of neural network usage is prediction of heavy metals recovery from wastewater [13]. Important problem related to neural modelling is highlighted in [14]. Final results of training and neural calculation can highly depend on initial values of weights (that are the most often selected randomly). For solution of this problem genetic algorithm was used.

In this application neural networks are used for calculation of chemical process efficiency, based on parameters and conditions of preparation. Modelling of such process can be treated as representation of data using function with one output and a few coefficients. This function and parameters are hard to simple definition, however based on experimental data representing specific behaviour, neural model of the chemical process can be prepared. For this purpose the feedforward neural networks with sigmoidal functions were selected for this task.

Adaptation of neural network for specified task is realized during training process. It concerns calculation of internal coefficients - weights. The most effective methods for this purpose are gradient-based optimization techniques. In this application the Levenberg-Marquardt algorithm was implemented. It is the second order optimization, the main point of this method is minimization of cost function using information about the Hessian matrix. It represents influence of weights changes on quality of network calculation [8]. Selected method used for coefficients of network selection give better results than algorithms based on gradients only. However it takes more time, but this disadvantage is not so much important during off-line training. One of the most important problem during design process of neural network is proper complexity of structure selection. Effective methods in real application is evolution of the network in parallel to adaptation of weights. So, at the same time values of neural network parameters and structure are modified [15]. Initial structure is defined a priori, then several connections are removed from the neural network. For correct determination of unnecessary connection additional saliency parameters (for each weights) are calculated. In application presented in this paper, decision is based on Optimal Brain Damage algorithm [16].

This paper is divided in five sections. Firstly, short introduction is presented. Then is the oil agglomeration process briefly described. In following part of article the structure of implemented neural network and method of training are shown. After presentation and analysis of results concluding remark are given.

2. Oil agglomeration process description

Oil agglomeration is a colloidal method, in which fine particles can be easily separated from mineral suspension via addition of oil to minerals slurry. A rapid mixing of such system leads to the adhesion of oil droplets into the mineral surface, where the formed liquid bridges link individual particles into larger aggregates, called agglomerates [17]. The success of the process strongly depends on the properties of the solid surface and oil/water interface. To be agglomerated the particles have to possess hydrophobic surface, in the case of hydrophilic ones after addition of oil the particles will remain un-agglomerated. This behaviour is very important especially during minerals processing, when a large amount of fine particles is produced via grinding of ore rock to reveal valuable minerals. Recovering of those valuable particles is a complex task because of the grains size (below 10 μm). Commonly used methods such as flotation, filtration or sedimentation in such condition are not very effective, and require additional steps. Oil agglomeration can be alternative for those methods, because the process makes possible to agglomerate the fine particles (below 5μm) with high selectivity using a simple equipment. Despite this, the process is not used on a large scale due to the high cost resulting from the use of large amounts of binding liquid (oil). Therefore, for several years, researchers have been trying to reduce the costs of the process. One solution is to introduce the binding liquid in a form of emulsion into the suspension [18]. It is happened because with the reduction of droplets size, the surface area increased, which leads to creation of more bridges on the mineral surface using small amount of oil. Moreover, the addition of surfactants to the emulsion lowers the interfacial tension resulting in formation of stronger agglomerates.

Modelling of such process is very difficult and time taking because it requires solving complex mathematical equations problematic in real applications. There is a lot of parameters that can affect the

course of oil agglomeration, such as oil type and amount, solid content, surfactant concentration, time, pH and agitation rate. That is way to avoid this difficulties Artificial Neural Network was utilized for modelling of the process. For preparation of the ANN model we used experimental data published in [18], where oil (kerosene) was introduced to the suspension in a form of emulsion with cationic surfactant. As mineral, dolomite was used, and the surface of the grains was modified with anionic collectors. During the experiment the pH, oil dosage, rate and time of mixing, and surfactants concentration were changed and taken for modelling as inputs. The output was process recovery calculated as ratio of mass agglomerate to total mass of the feed.

3. Mathematical description of neural model, training process and Optimal Brain Damage pruning

Neural network is combination of elementary nodes that realize multiplication of input signals, then achieved values are added, in the next stage the results are arguments for activation function. Individual neurons are connected in a structure that globally performs complex function (figure 1). Exemplary (for network with one hidden layer) equation describing details of ANN calculations is presented below:

$$\mathbf{y}_{\mathbf{nn}}(\mathbf{x}_{\mathbf{in}}) = f_o(\mathbf{h}) = f_o\left(b_0 + \sum_{j=1}^{J} w_j f_h\left(b_{0j} + \sum_{i=1}^{N} w_{ij} x_i\right)\right) \tag{1}$$

where: $\boldsymbol{y_{nn}}$-vector of output values, $\boldsymbol{x_{in}}$-input vector, $\boldsymbol{h}$-vector of arguments for activation function in output layer, N-number of inputs, J-number of nodes in hidden layer, w_{ij}-weights between i-th and j-th node, b_0-bias values, f_o-activation function of the output layer (the most often linear function), f_h-activation function of the hidden layer (the most often sigmoidal function).

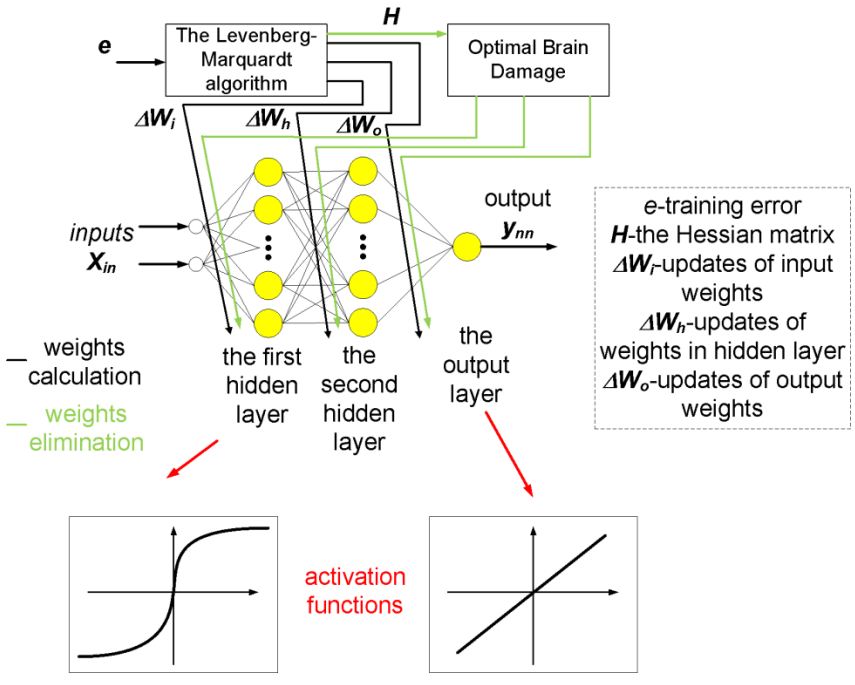

Figure 1. The topology of applied Artificial Neural Network.

One of the main elements affecting the final value of neural structure y_{nn} are values of weight coefficients. Therefore, for correct approximation of elements not included in training dataset, optimal selection of those parameters is particularly important. In this application, for this purpose the Levenberg-Marquardt algorithm was implemented [15]. In each iteration k of this method, weights are recalculated, introducing correction $\Delta \boldsymbol{w}$:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \Delta\mathbf{w}(k). \tag{2}$$

Training process assumes modification of weights in order to minimization of processing error - cost function ***E***. Simplified definition of this value can be expressed as (assuming one output of neural model):

$$\mathbf{e}(\mathbf{w}) = \left[y_{nni}(\mathbf{w}) - d_i \right], \tag{3}$$

where: d_i – *i*-th reference value, y_{nni} – *i*-th neural network output value (obtained for d_i). The base information during calculation of weights is the Jacobian matrix, defined using following expression (for *n* weights and *M* patterns):

$$\mathbf{J}(\mathbf{w}) = \begin{bmatrix} \frac{\partial e_1}{\partial w_1} & \frac{\partial e_1}{\partial w_2} & \cdots & \frac{\partial e_1}{\partial w_n} \\ \frac{\partial e_2}{\partial w_1} & \frac{\partial e_2}{\partial w_2} & \cdots & \frac{\partial e_2}{\partial w_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_M}{\partial w_1} & \frac{\partial e_M}{\partial w_2} & \cdots & \frac{\partial e_M}{\partial w_n} \end{bmatrix}. \tag{4}$$

According to the Gauss-Newton equation, correction of weights are calculated using formula presented below:

$$\Delta\mathbf{w} = -\left(\mathbf{J}(w)^T \mathbf{J}(\mathbf{w})\right)^{-1} \mathbf{J}^T(\mathbf{w})\mathbf{e}(\mathbf{w}). \tag{5}$$

In equation (5) the Hessian matrix ***H*** is defined using following equation:

$$\mathbf{H}(\mathbf{w}) = \left(\mathbf{J}(w)^T \mathbf{J}(\mathbf{w})\right), \tag{6}$$

then, combining (5) and (6):

$$\Delta\mathbf{w} = -\left(\mathbf{H}(w)\right)^{-1} \mathbf{J}^T(\mathbf{w})\mathbf{e}(\mathbf{w}). \tag{7}$$

For simplification of inverse matrix $\boldsymbol{H}^{-1}$ calculation, approximation of ***H*** is introduced:

$$\mathbf{G}(\mathbf{w}) = \mathbf{H}(\mathbf{w}) + \mu\mathbf{I}\ . \tag{8}$$

where: ***I***-the identity matrix. The inverse of the matrix ***G*** is possible to calculate if:

$$(\lambda_i + \mu) > \mathbf{0}\ , \tag{9}$$

where: λ_i – i-th element of the eigenvalue vector of ***H*** matrix. Concluding above presented considerations, update of network coefficients are calculated using following expression:

$$\Delta\mathbf{w} = -\left(\mathbf{J}(w)^T \mathbf{J}(\mathbf{w}) + \mu\mathbf{I}\right)^{-1} \mathbf{J}^T(\mathbf{w})\mathbf{e}(\mathbf{w})\ . \tag{10}$$

In presented work, the Levenberg-Marquardt algorithm was combined with Optimal Brain Damage method implemented for weights elimination. The OBD defines coefficients calculated for each parameter of neural network:

$$S_i = \Delta E = \frac{1}{2}\sum_i h_{ii}[\Delta w_{ii}]^2 \tag{11}$$

The main information in OBD is also the hessian matrix, it should be mentioned that the ***H*** matrix is diagonally dominant, so only diagonal elements are used in equation (11). Details are presented in [17]. Based on this information network can be reduced (coefficients with the smallest saliency values S_i). It should be noted that some information used for calculation of *Si* index can be taken after the Levenberg-Marquardt algorithm processing.

Following stages of OBD algorithm operation are presented in figure 2. Algorithm starts with predefinition of initial parameters: initial structure of model, number of weights removed in each sequence, number of all removed weights, conditions of training, stopping criteria, etc. Firstly, complete training is realized (according to (10)). Then, selection of weights designed for elimination should be done. The decision about replacing actual value to zero is made based on value of coefficients S_i (calculated for several coefficients of neural network). In following step, weights are eliminated and overall error is calculated. If closing conditions are not achieved, the whole process is repeated.

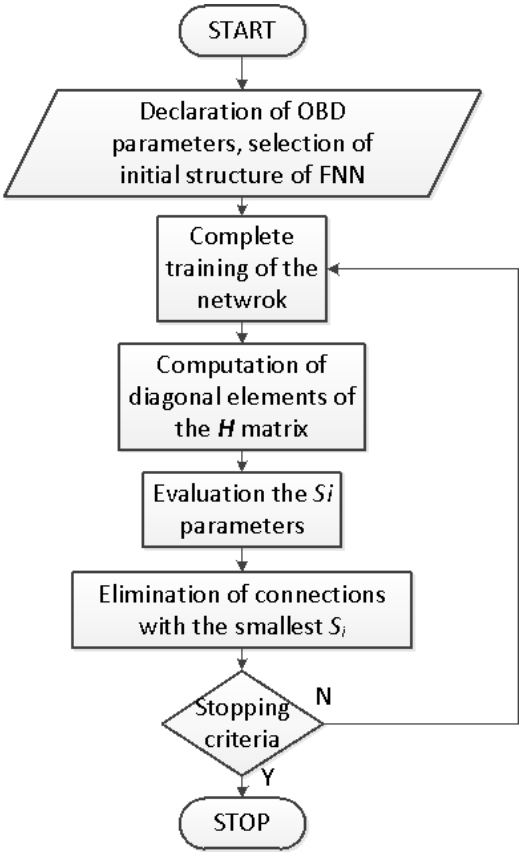

Figure 2. Flowchart presenting data processing in OBD algorithm.

4. Results

Based on theoretical description of neural model and training process, including structure optimization, presented in previous parts of this article, special code was prepared. Then tests were done. The database was prepared in chemical laboratory, so input vector of ANN contains (respectively): mixing rate, time of mixing, anionic surfactant amount (mg/g), cationic surfactant amount (mg/g), kerosene amount (ml/g), pH. As anionic and cationic surfactant used sodium oleate and dodecylammonium hydrochloride, respectively, which are commonly used in mineral processing as collectors. Output of the neural network model is recovery. The number of samples (input-output elements) included in training vector was equal 52.

The first research presents general trend of error fluctuation under changes of neural network topology (figure 3). Only exemplary tests are shown, other conditions of training were assumed. However the goal was observation of overall tendency of error changes. Each value was calculated using following formula:

$$Err = \frac{1}{M}\sum_{i=1}^{M}\left|y_{nni} - d_i\right|. \tag{12}$$

In following iteration of OBD algorithm, the connections are removed (weights set to zero). It can be observed that initial number of hidden nodes was not selected properly. After elimination of some group of weights, quality of calculation was improved (optimal structure was obtained). This is in line with theory of neural networks.

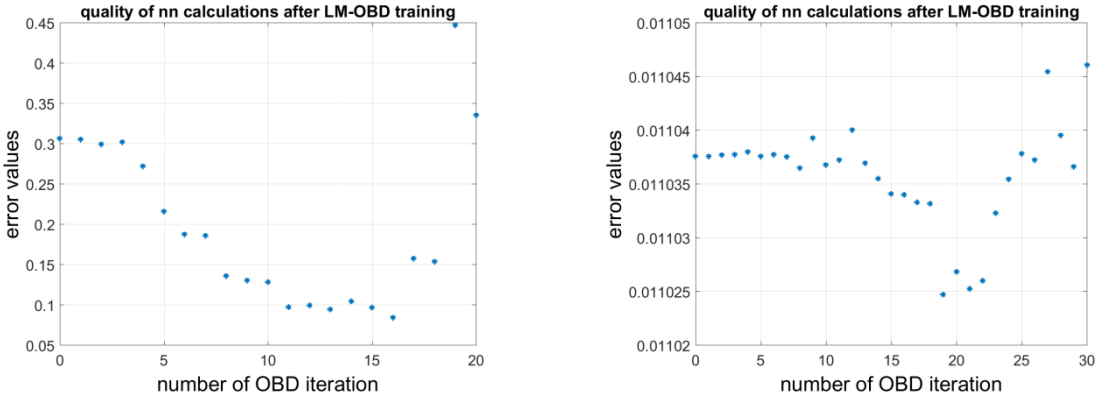

Figure 3. Exemplary values of processing errors during connection reduction.

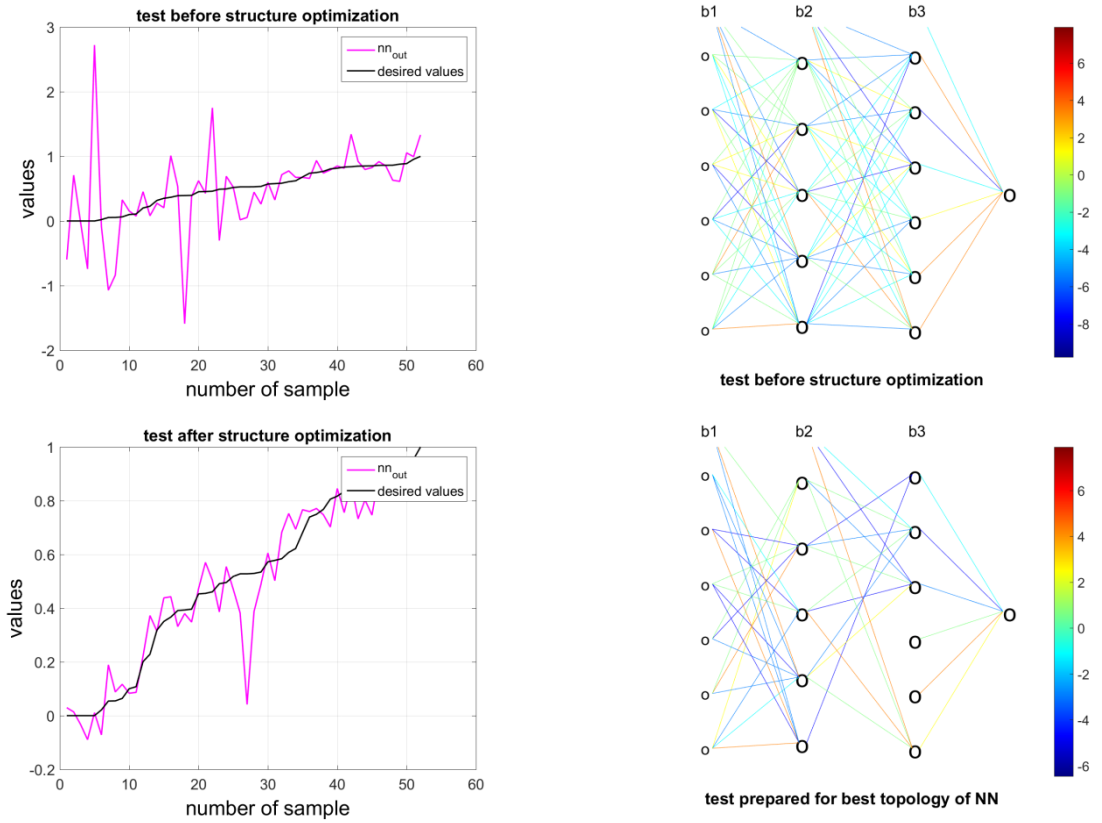

Figure 4. Tests presenting influence of structure optimization on recovery calculation.

Subsequent pictures (figure 4) show details of exemplary training process and results of structure optimization. Next to the results, presenting real and calculated values of recovery, in the right column changes of neural topology is shown. The colours of lines responds to values of weights. Initial structure was {6-5-6-1}, it means: 6 inputs, 5 neurons in the first hidden layer and 6 nodes in the second layer, 1 output. The whole OBD optimization, in this test, was repeated 30 times, in each iteration 3 connections were reduced. Best results were obtained after removing 33 links. However, it should be mentioned that, in the best topology whole nodes were eliminated. Implemented optimization method modifies the ANN, it gives significant improvements of results.

One of important expectation for neural networks is high quality of processing, under input values not included in training process. In the presented work, training data was not used in the tests. Generally, it is possible to divide samples into two groups, but in the case of the task being analyzed this is problematic due to the use of experimental data (limited number of samples). For this reason, in all tests, the measured values were appropriately disturbed by introducing noise.

Best results achieved in this research are presented in figure 5. The LM-OBD algorithm has started with following neural network: {6-15-16-1}, number of all removed connections was equal 60, number of removed connections for the best results was equal 57. It should be also noted that each training using the Levenberg-Marquardt method takes 1000 iteration (this number was selected experimentally, for dataset used in this project, based on observation of validation error and gradient), and total time of calculations using LM-OBD was 22.3157s.

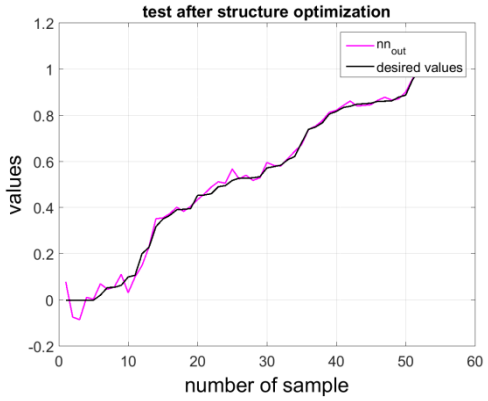

Figure 5. Example of ANN calculation – the best results.

5. Conclusions

This paper is focused on dolomite particles oil agglomeration modelling using neural networks. Based on obtained neural model, recovery of the process, for introduced input parameters, can be estimated. In next step, optimization of the process, based on prepared neural model can be done (assessment of the required input parameters in order to achieve the assumed performance). Precision of calculation of obtained neural model was very high. It is worth to mention that all calculations were done for real data collected in laboratory experiment. Important part of this work includes optimization of the ANN's structure. For this purpose Optimal Brain Damage method was implemented. Results confirm theoretical considerations related to relationship of neural network complexity and results of processing.

Acknowledgments

This work was financially supported by a statutory activity subsidy from the Polish Ministry of Science and Higher Education for the Faculty of Chemistry (A.B.) and for the Faculty of Electrical Engineering (M.K.) of Wroclaw University of Science and Technology.

6. References

[1] Nowopolski K, Wicher B, Łuczak D and Siwek P 2017 *Proc. Int. Conf. on Methods and Models in Automation and Robotics (Miedzyzdroje)*, p 576
[2] Pajchrowski K, Zawirski K and Nowopolski K 2015 *IEEE T. Ind. Inform*. **11(2)** p 560
[3] Cai JP, Xing L and Zhang M 2017 *IEEE Access* **5** p 15839
[4] Merabet A, Tanvir AA and Beddek K 2016 *IET Renew. Power Gen.* **10(10)** p 1597
[5] Khomfoi S and Tolbert LM 2007 *IEEE Trans. Power Electron* **22(3)** p 1062
[6] Chen CI and Chen YC 2015 *IEEE Trans. Power Del.* **30(3)** p 1577
[7] Yoo B, Kwak Y, Kim Y, Choi C and Kim J 2018 IEEE Signal Process. Lett. **25(6)** p 808
[8] Bishop CM 1995 *Neural Networks for Pattern Recognition* (Oxford: Oxford University Press)
[9] Mandic D and Chambers J 2001 *Recurrent Neutral Networks for prediction: learning, algorithms, architectures and stability*, 1st edition (USA: Wiley)
[10] Lawrence S, Chung Tsoi A and Back AD 1996 *Proc. Australian Conf. on Neural Networks* p 16
[11] Kalyani VK, Pallavika, Sanjay Choudhuri, Gouri Charan T, Haldar DD, Kamal KP, Badhe YP, Tambe SS and Kulkarni BD 2007 *Min. Proc. Ext. Met. Rev.* **29(2)** p 130
[12] Nakhaei F and Irannaiad M 2013 *Physicochem. Probl. Miner. Process.* **49(1)** p 255
[13] Allahkarami E, Igder A, Fazlavi A and Rezal B 2017 *Physicochem. Probl. Miner. Process.* **53(2)** p 1105
[14] Allahkarami E, Nuri OS, Abdollahzadeh A and Rezal B 2017 *Physicochem. Probl. Miner. Process.* **53(1)** p 366.
[15] Hassibi B, Stork DG and Wolff GJ 1992 Optimal brain surgeon and general network pruning *IEEE Int. conf. on neural Networks* **1** p 293

[16] Le Cun Y, Denker JS and Solla SA 1990 *Adv. Neural Inform. Proces. Syst.* **2** p 598
[17] Cebeci Y and Sönmez I 2004 *J. Colloid Interface Sci*. **273(1)** p 300
[18] Bastrzyk A, Polowczyk I., Sadowski Z. and Sikora A 2011 *Sep. Purif. Technol.* **77(3)** p 325