NÁVRH ADAPTIVNÍ UMĚLÉ NEURONOVÉ SÍTĚ A JEJÍ POUŽITÍ PŘI KALIBROVÁNÍ NUMERICKÝCH MODELŮ

SELF-ADAPTIVE ARTIFICIAL NEURAL NETWORK IN NUMERICAL MODELS CALIBRATION

Soutěžní práce

Autor:

TOMÁŠ MAREŠ

Odborné vedení:

Ing. Anna Kučerová, Ph.D.

SOUTĚŽ O CENU AKADEMIKA BAŽANTA

6. května 2010

Abstrakt

Navzdory rostoucímu výpočetnímu výkonu moderních počítačů a počítačových sítí jsou simulace stavebních konstrukcí stále časově velmi náročné. Proto je velmi výhodné využívat vhodné aproximace některých zdlouhavých výpočtů, a to zejména v oblasti identifikace parametrů materiálových a konstrukčních modelů. Vrstevnaté neuronové sítě představují velmi robustní aproximační nástroj. Jejich praktické využití není však příliš jednoduché, protože je třeba správně zvolit vhodnou topologii sítě: počet vrstev a počet jejich neuronů. Předkládaná práce se věnuje návrhu jednoduché neuronové sítě, která si při procesu učení sama určí nejvýhodnější počet neuronů ve skryté vrstvě. Aproximační schopnosti navržené sítě jsou testovány na několika matematických problémech a při identifikaci parametrů mikroploškového modelu m4. Identifikace je provedena dvěma různými způsoby. V první variantě je neuronová síť použita pro aproximaci numerického modelu pro rychlý odhad odezvy modelu na základě daných materiálových parametrů a zatížení. V druhé variantě je použita pro konstrukci inverzního modelu, který je schopen předpovědět hodnoty materiálových parametrů pro danou (naměřenou) odezvu numerického modelu.

Abstract

Despite the growing performance of modern computers and clusters, a suitable approximation of an exhaustive simulation of structural models has still many applications in engineering problems. For example, the field of parameters identification may represent the largest domain for very efficient applications. The layered neural networks are considered as very general tools for approximation. The practical usage is, however, non-trivial in the choice of an appropriate architecture. The presented contribution is concerned with the development of a simple neural network with the self-adaptive architecture. Its approximation abilities are tested on several mathematical problems and two different modes of material parameters' identification problem. In the first one, the neural network is used to approximate the numerical model predicting the response for a given set of material parameters and loading. The second mode employs the neural network for constructing an inverse model, where material parameters are directly predicted for a given response.

1 Introduction

A variety of engineering tasks nowadays lead to an inverse analysis problem. Generally, the aim of an inverse analysis is to rediscover unknown inputs from the known outputs. In common engineering applications, a goal is to determine the initial conditions and properties from physical experiments or, equivalently, to find a set of parameters for a numerical model describing properly the experiment.

While the numerical model of an experiment represent a well-defined mapping from input (model, material, structural, or other) parameters to output (structural response), there is no guarantee that the inverse relation even exist. In engineering practice is the inverse relation often ill-posed, highly nonlinear and multi-modal. Therefore, the choice of an appropriate identification strategy is not trivial. Moreover, such identification process is supposed to be performed repeatedly for any new measurement and therefore, the emphasis is also put on the efficiency of chosen identification method.

In overall, there are two main philosophies to solution of identification problems. A forward (classical) mode/direction is based on the definition of an error function of the difference between outputs of the model and experimental measurements. A solution comes with the minimum of this function. This mode of identification could be considered as more general and robust, but repeated application is relatively computationally expensive. The second philosophy, an inverse mode, assumes the existence of an inverse relationship between outputs and inputs. If such relationship is established, then the retrieval of desired inputs is a matter of seconds and could be easily executed repeatedly. For a more interested reader about identification strategies, see the contribution of Kučerová and Lepš in this proceedings.

Artificial neural networks (ANN) [2], [4] are powerful computational systems consisting of many simple processing elements – so-called neurons – connected together to perform tasks analogously to biological brains. Their main feature is ability to change their structure based on external information that flows through the ANN during the learning (training) phase.

A particular type of ANN is so-called feedforward neural network, which consists of neurons organized into layers where outputs from one layer are used as inputs into the following layer. There are no cycles or loops in the network, no feed-back connections. Mostly used example is a multi-layer perceptron (MLP) with a sigmoid transfer function and gradient descent method of training called back-propagation learning algorithm. In practical usage, MLP are known for their ability to approximate non-linear relations and therefore, when speaking about ANN, particularly MLP are considered in the following text.

In the field of parameter identification, ANN has two main applications. In the forward mode of identification, ANN can be used to approximate the computationally expensive numerical model. Approximation of numerical model can be found relatively easily, since such relation is well-posed. ANN can be then efficiently used in the phase of parameter optimization where the huge number numerical model evaluations are replaced by very fast evaluations of ANN. The only drawback of such application is often high number of outputs. Numerical mechanical models have usually a small number of input parameters describing the material or structure. Nevertheless, there is typically huge number of outputs, for example the load-deflection curve defined in dozens of discrete points. In this case, one search either for one ANN with many outputs or many simpler ANNs with one output each.

In the inverse mode of identification, the ANN can be applied to approximate the inverse relation between inputs and outputs. Search for this relation can be non-trivial for the reasons mentioned above, but once such relation is found, it can be very quickly and repeatedly used for estimating parameters from any new experiment and no other optimization process is necessary.

When dealing with ANNs, the key point is the choice of its architecture. The number of units in input and output layer is usually given, but it remains to decide the number of units in hidden layer. In this contribution, we present a simple self-adaptive

ANN, which automatically determine an optimal number of hidden neurons. The resulting algorithm is tested on several mathematical problems and finally applied to parameters identification of microplane model M4 [1]. The forward and inverse strategy is compared.

2 Architecture of Artificial Neural Network

To introduce a reader into the problematic of ANN, let's recapitulate its main principles. A multi-layer feedforward neural network is a particular ANN, where processing units are organized into parallel layers, see Fig. 1(a).

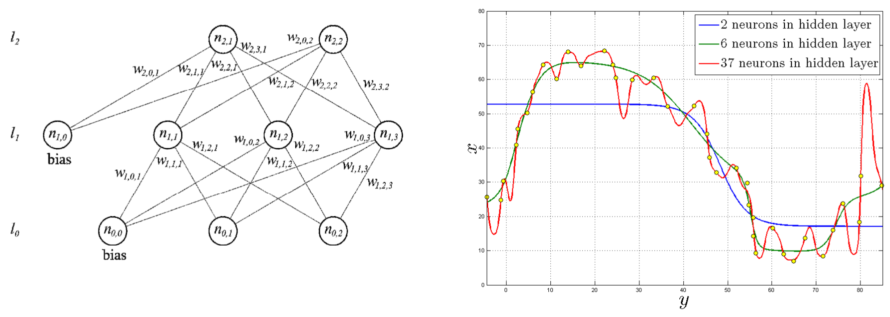

Fig. 1. (a) Architecture of an multi-layer feedforward neural network; (b) Underfitting and overfitting of ANN

The input layer represents directly a vector of input parameters. These values are then multiplied by a vector of constants, so-called synaptic weights and summarized. The result is then used as an input into the units of the following, so-called hidden layer. Each element in the hidden layer – neuron – is defined by an activation function, which is applied on the input and produces the output value of the neuron. The output is then again multiplied by other synaptic weights and again used as input in next layer and so on. The synaptic weights are parameters of ANN to be determined during the training process. The type of activation function is usually chosen in accordance with the type of function to be approximated. In the case of continuous problems, sigmoid activation functions seem to be most appropriate.

Into input and hidden layers, one bias neuron is also added. It doesn't consist of activation function, but only a constant value. Its role is to enable to shift the value of sum over the outputs of his neighbouring neurons before this sum enters as input into the neurons in the following layer. The value of biases is determined by training process together with synaptic weights.

Despite of ANN's popularity there are only few recommendations for the choice of ANN's architecture. The authors, e.g. in [5], [6], shows that ANN with any of a wide variety of continuous nonlinear hidden-layer activation functions, one hidden layer with an arbitrarily large number of units suffices for the "universal approximation" property. Therefore, we limit our numerical experiments to such as case. But there is no theory yet to decide how many hidden units are needed to approximate any given function.

In general the choice of number of hidden units depends on many factors such as the number of input and output units, the number of training samples, the complexity of the function to be approximated, the type of hidden unit activation function, noise in the target values etc. In [3] it is indicated that the number of training samples NTR should be larger than the number of adjustable ANN's parameters. It implies that

$$NH < \frac{NTR-1}{NI+2}, \tag{1}$$

where *NH* is the number of hidden units and *NI* is the number of inputs. The inequality (1) creates an upper bound for the number of hidden units, but this limit value is usually far from optimal. When looking for some better value of *NH*, the choice should be driven by following principles:

i. If ANN produces high error on both the training and testing data due to so-called *underfitting*, ANN's architecture is probably too simple and more hidden units should be added.
ii. If ANN produces relatively small error on training data, but in orders higher error on testing due to *overfitting*, there are probably too many hidden units and some of them should be eliminated.

Regarding these principles, demonstrated also in Fig. 1(b), we have developed a simple ANN with the ability to adapt the number of hidden neurons. The algorithm is following:

- The ANN starts with one hidden neuron.
- The process of ANN's training is executed.
- At the end, we compute the average absolute error on training data *ETR* and testing data *ETE.*
- New hidden neuron is added into the ANN if the condition (1) is fulfilled together with the following condition

  $$\frac{ETE}{ETR} < TTER, \tag{2}$$

  where *TTER* is a given testing to training error ratio. These two conditions we call adaptivity criteria.
- With the new architecture, new training process is executed.
- If one of those conditions is not fulfilled, the process of adding hidden neurons is stopped and the last used architecture and synaptic weights are stored as the best result.

In other words, the algorithm starts with an underfitted ANN and proceeds by adding hidden neurons until the ANN is overfitted. The criterion of overfitting is based on the proportion of testing to training error. Since the high error on testing data together with small error on training data defines the overfitting, we stop the process of adding neurons, when the error on testing data exceeds the value of *TTER*, defining a certain multiple of the error on training data. In order to choose an appropriate value of TTER, we have performed the following numerical study.

3 Numerical Study of ANN's Adaptivity Features

We have decided to test the abilities of ANN on a simple example, where the ANN should approximate as well as possible highly non-linear relation given as

$$F(x) = \sum_{i=1}^{6} i \sin((i+1)x + i), \tag{3}$$

which is shown in Fig. 2(a).

We have generated 100 training and 100 testing pairs {x; F(x)} from the uniform distribution over the domain of $x \in [-5;5]$. As a training algorithm, an enhanced variant of popular backpropagation algorithm, so-called *Resilient backpropagation*, is implemented. It is a local adaptive learning scheme, performing supervised batch learning in multi-layer perceptrons. For a detailed discussion see e.g. [9]. The algorithm has three adjustable parameters concerning the steps of changing synaptic weights: the initial update-value Δ_0, the maximum step size Δ_{max} and the minimum step size Δ_{min}. The values of these parameters are set to $\Delta_0 = 0.001$, $\Delta_{max} = 10$ and $\Delta_{min} = 10^{-8}$. The other parameters are considered as constants with values recommended by authors.

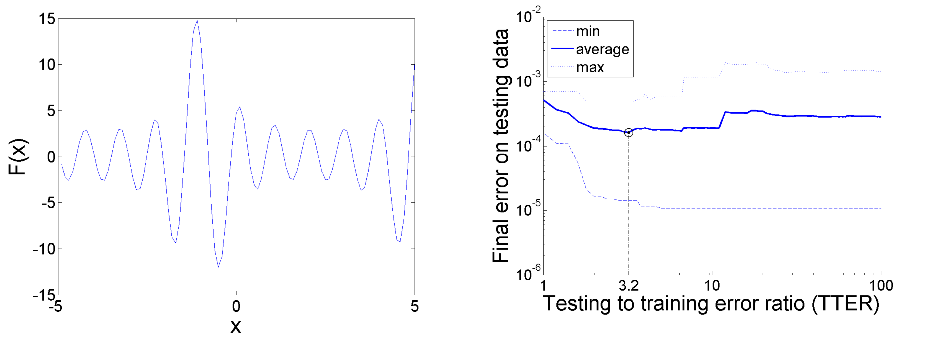

Fig. 2. (a) Relation to be learned by ANN; (b) Relation between the resulting error on testing data *ETE* and value of testing to training error ratio *TTER*.

The training process starts with initial values of synaptic weights chosen randomly from a set of three values {-1; 0; 1}. The training continues by 500 iterations and the resulting ANN's configuration is stored. Such starting training is performed 50-times, every time with new random initial values of synaptic weights. Among 50 resulting configurations, the one with minimal average absolute error on training data is chosen. For this ANN, the second phase of training is performed, starting from the stored configuration and proceeding in new iterations. This phase is ended when one of following criteria is fulfilled:

- total number of iterations exceeds 10,000;
- $$\frac{ETR_{k-1} - ETR_k}{ETR_{k-1}} < 0.0001, \tag{4}$$

where ETR_k is average absolute error on training data in actual iteration k and ETR_{k-1} is the same error computed in previous iteration *k-1*. The second criterion is applied to stop the training, when the convergence becomes too slow – the error on training data almost is almost not changing. It indicates that some local extreme is located and no other iterations are needed. When the training process is finished, the adaptivity criteria are tested and eventually new hidden neuron is added and new training process is launched. Adding of new hidden neuron leads to creation of new synaptic weights. The following training process uses as a start point the resulting values of existing synaptic weights from the previous training and random values for new synaptic weights. This is repeated until the adaptivity criteria are fulfilled.

To establish an appropriate value of *TTER*, we have performed 100 independent training processes with the adaptivity, but without the adaptivity condition (2). This condition was applied on the obtained data post facto and the value of *TTER* was varied from 1 to 100. The goal was to find the value of *TTER*, for which the resulting error on training data *ETE* is minimal. The results of this study are plotted in Fig. 2(b) and regarding them, the value *TTER* = 3.2 was chosen as optimal.

We were also interested, whether the sequential changes of synaptic weights have some influence on the final error on testing data. We compared our results with other computations, where the adaptivity is used to find an optimal architecture in the first step and then, all synaptic weights are randomized and new training process is launched to establish their values on a given fixed topology. In other words, all the information about synaptic weighs obtained during the adaptation process is forgotten. The resulting errors on testing data obtained for 100 independent computations by both strategies are shown in Fig. 3(a) and one can see that the information during the adaptation process can significantly decrease the resulting error in ANN's predictions.

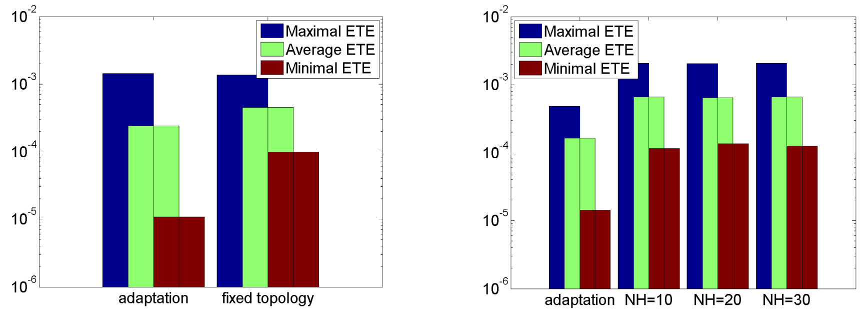

Fig. 3. (a) The error on testing data after adaptation and in case of training on the optimal fixed topology; (b) The error on testing data after adaptation and in case of training on arbitrarily chosen fixed topology.

The most common way for establishing the number of hidden units is the simplest trial-and-error method. Having fixed number of training samples, we can use the rule (1), which tells us that number of hidden units should be smaller than 33. To have an idea about the results of an trial-and-error method in comparison with the results from adaptivity, we have trained ANN's with fixed topology for three different numbers of hidden neurons: NH = 10, 20 and 30. The final error on testing data computed over 100 independent runs of training process is depicted in Fig. 3(b).

We should also note that the adaptivity criteria lead to a stochastic solution. Every run of adaptivity process can terminate with a different optimal architecture. On the particular problem studied here, the histogram in Fig. 4(a) was obtained for number of hidden neurons over 100 runs of adaptation process.

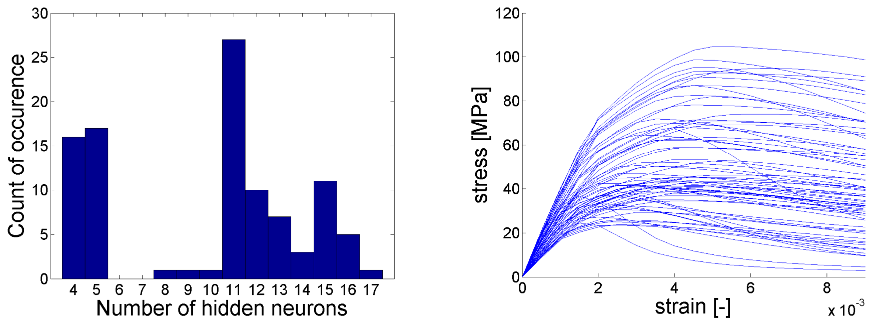

Fig. 4. (a) Histogram of number of hidden neurons over 100 runs of adaptation process; (b) Bundle of stress-strain diagrams generate for training and testing of ANN

4 Application of ANN in parameters identification of microplane model M4

Concrete is a heterogeneous material and therefore the simulation of its behaviour encounters serious difficulties, both theoretical and numerical. The microplane model M4 [1] is a fully three-dimensional material law that includes tensional and compressive softening, damage of the material, different combinations of loading, unloading and cyclic loading. It can describe the development of anisotropy within the material. The major disadvantage of this model, however, is an enormous computational cost associated with structural analysis and phenomenological material parameters without clear physical interpretation. Therefore, a reliable procedure for parameters identification is on demand. Some work on parameters estimation of microplane model was already presented, e.g. in [7], [8]. Here, we would like to present two possible applications of ANN in parameters identification of microplane model and discuss their advantages and drawbacks.

Because of the limited space for this contribution, we focus on identification of three parameters – Young's modulus E, k_1 and c_{20} – which should be identified from uniaxial compression test. When simulating uniaxial compression test, the model output is a stress-strain diagram. Particularly, we simulate uniaxial compression of a concrete cylinder with diameter equal to 15cm and height equal to 30cm. We discretize the stress-strain diagram into 18 discrete points corresponding to fixed values of strain and corresponding 18 values of stress $\sigma_1, ..., \sigma_{18}$ are considered as model outputs. Because of high computational demands of each compression test simulation, only 60 samples were generated for a training set and 10 samples for a testing set. The resulting bundle of stress-strain diagram is shown in Fig. 4(b).

We start by the inverse mode of identification where ANN is supposed to approximate the inverse relation between model outputs and model parameters (here, considered as inputs) listed above. Since the neural network will be trained to approximate the inverse relation, 18 values of stress becomes ANN's inputs. These 18 values are, however highly correlated and therefore, only several important values are chosen among them to be used as inputs. The choice can be driven by *Pearson product-moment correlation coefficient*, which can be computed for pairs consisting of one stress value and one parameter. The computed values of correlation are presented e.g. in [7]. To simplify more the training process, one ANN is trained with adaptivity for each model parameter. The set of inputs and resulting architecture is described in Table 1.

Table 1. Inverse mode – architecture of ANNs.

Parameter	Inputs	Architecture
E	σ_1, σ_2, σ_3	3 – 5 – 1
k_1	σ_5, σ_{18}, σ_{peak}, σ_{peak}, $E_{\text{prediction}}$	5 – 4 – 1
c_{20}	σ_6, σ_8, σ_{12}, σ_{16}, $E_{\text{predict.}}$, $k_{I,\text{predict.}}$	6 – 3 – 1

For a judgement of ANN performance we computed the average error on training and testing data relative to the range between minimal and maximal values of each model parameter used in training and testing sets. The resulting relative errors are presented in Table 2. One can see that ANN can very precisely found the inverse relation for prediction Young's modulus and parameter k_1, but it is unable to approximate the inverse relation for parameter c_{20} with satisfactory precision. So the application of ANN in the inverse mode is not always trivial.

Table 2. Inverse mode – resulting average relative error on training and testing data.

Parameter	Av. Relative *ETR* [%]	Av. Relative *ETE* [%]
E	0.18	0.34
k_1	0.46	0.86
c_{20}	10.44	22.43

In the case of forward mode, the ANN can be used for the approximation of the numerical model itself. In that case, however, there is a relatively small number of ANN's inputs – only four model parameters (Young's modulus E, k_1 and c_{20} parameters must be accompanied also by Poison's ratio, which cannot be identified only from axial deformation, but has still an indispensable influence on its shape). But there is a larger number of outputs corresponding to discrete points of stress-strain diagram. In order to predict stress values in these points, there are two possibilities of ANN implementation.

In the first scenario, one independent ANN can be trained to predict the stress in one chosen point. Such ANN can be very simple, training process can be also fast and easy, but we must train 18 different ANNs. Fig. 5(a) shows the minimal, average and maximal error on training as well as on testing data for each ANN, which predicts the stress value in one of 18 points.

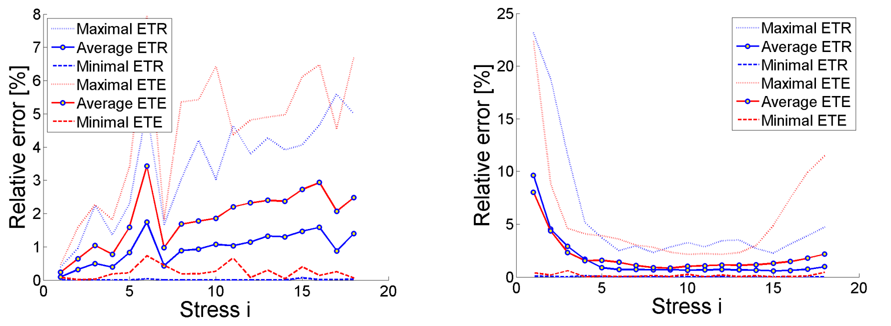

Fig. 5. (a) Relative errors in predictions of 18 simple ANNs; (b) Relative errors in prediction of one complex ANN.

The resulting number of hidden neurons of each ANN was $NH = [4; 7]$. One can see that average error on both the training and testing data of all ANNs is smaller than 4% and worst cases have not exceeded the error of 8%. This precision we consider as satisfactory. The only disadvantage of this approach remains the necessity of training number of independent ANNs. If we would like to approximate the stress-strain diagram in more discrete points for a better accuracy, the training process becomes lengthier. Moreover we should recall that training of ANNs is only a first phase of the identification process which proceeds in optimization of model parameters in order to fit the response of all ANNs to experimental data.

In the second scenario, only one ANN can be trained, if we add the value of strain as the fifth input. 60 training diagram consisting of 18 points change to 1080 training samples and 10 testing diagrams create 180 testing samples. The relation to be approximated becomes of course more complicated, which is coherent with resulting number of hidden neurons $NH = 11$. The resulting errors on training and testing data, listed in Table 3, are expressed relatively to bounds for the output stress which is quite large, because it embodies all stress values during the whole loading process.

Table 3. Forward mode – resulting relative error on training and testing data.

Error	Minimal [%]	Average [%]	Maximal [%]
ETR	0.0010	0.7234	4.4428
ETE	0.0157	1.1017	10.7873

Nevertheless, one can be also interested in the accuracy of ANN's prediction in particular discrete points of the diagram and how the errors become large when computed relatively to bounds of stresses corresponding to these particular points. Such error distribution is depicted in Fig. 5(b). One can see that in the beginning of the loading process where the values of stress are small, the relative error becomes high because the stress bounds are very narrow. In the middle of the loading, the stresses are high and the error relatively to them small, and at the end of the loading the stress value again decreases and the relative error increases. Nevertheless, the average relative error exceeds 5% only in the first point of diagram and its satisfactory small in the rest. When comparing to first scenario of forward mode, the errors are in general higher, but the usage of one ANN is of course simpler.

5 Conclusions

In the presented contribution, we focus on application of artificial neural networks in parameters identification. An easy implementation of feedforward multi-layer neural network is complicated by non-trivial choice of ANN's architecture, especially the number of neurons in hidden layer. Here, we propose a simple algorithm for an automatic determination of hidden neurons. Moreover, results in Fig. 3(a) shows that the complete process of sequential training accompanied by adaptivity of hidden layer leads to better results than the simple training of well chosen architecture.

The second part of the paper concerns the application of the resulting algorithm for ANN training to parameters identification of microplane model m4. Three different scenarios are demonstrated and their particular advantages and drawbacks are discussed throughout the paper in very detail.

Acknowledgement. The financial support of this work by research project MSM6840770003 and by MPO project FT-TA4/100 is gratefully acknowledged.

References

1. Z. P. Bažant, F. C. Caner, I. Carol, M. D. Adley, S. A. Akers, "Microplane model M4 for concrete. Part I: Formulation with work-conjugate deviatoric stress, Part II: Algorithm and calibration", Journal of Engineering Mechanics, vol. 126, no. 9, pp. 944-953, 954-961, 2000.
2. K. N. Gurney, An introduction to neural networks, UCL Press: London, 2002.
3. Hagan MT, Demuth HB, Beale M. Neural network design. PWS Publishing Company; 1996.
4. S. S. Haykin, Neural networks and learning machines, Prentice Hall/Pearson: New York, 3rd edition, 2009.
5. K. Hornik, M. Stinchcombe, H. White, "Multilayer feedforward networks are universal approximators", Neural Networks, vol. 2, pp. 359-366, 1989.
6. K. Hornik, "Some new results on neural network approximation", Neural Networks, vol. 6, pp. 1069-1072, 1993.
7. A. Kučerová, Identification of nonlinear mechanical model parameters based on softcomputing methods, Ph.D. thesis, Ecole Normale Supérieure de Cachan, Laboratoire de Mécanique et Technologie, 2007.
8. Kučerová, M. Lepš and J. Zeman, "Back analysis of microplane model parameters using soft computing methods", CAMES: Computer Assisted Mechanics and Engineering Sciences, vol. 14, no. 2, pp. 219-242, 2007.
9. M. Riedmiller, "Advanced supervised learning in multi-layer perceptrons — From backpropagation to adaptive learning algorithms", Computer Standards & Interfaces, vol. 16, no. 3, pp. 265-278, 1994.