Taylor & Francis
Taylor & Francis Group

Statyba

ISSN: 1392-1525 (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tcem19

NEURAL NETWORK MATERIAL MODELLING

J. Ghaboussi , X. Wu & G. Kaklauskas PhD

To cite this article: J. Ghaboussi , X. Wu & G. Kaklauskas PhD (1999) NEURAL NETWORK MATERIAL MODELLING, Statyba, 5:4, 250-257, DOI: 10.1080/13921525.1999.10531472

To link to this article: https://doi.org/10.1080/13921525.1999.10531472

Published online: 26 Jul 2012.

Submit your article to this journal

Article views: 200

Citing articles: 2 View citing articles

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=tcem20

ISSN 1392-1525. STATYBA - CIVIL ENGINEERING - 1999, V tomas, Nr. 4

NEURAL NETWORK MATERIAL MODELLING

J. Ghaboussi, X. Wu and G. Kaklauskas

1. Introduction

A constitutive law or a material model is conventionally described as a mathematical model represented as stress-strain relations that convey human perception of the behaviour of a material. In engineering mechanics, material modelling constitutes an integral part in the study of the structural behaviour under external excitation. With the availability of powerful computing machines and sophisticated computational methods, such as the finite element method, and the advances in experimental instrumentation and testing methods, the importance of the role that material modelling plays in computational mechanics is greatly enhanced. On the other hand, with the introduction of modern composite materials, the constitutive modelling of their complex behaviour becomes increasingly more involved.

Recent advances in neural networks, especially the new insights in developed learning algorithms, have facilitated the development of a fundamentally different approach to material modelling using neural networks. Within the framework of information science, the constitutive modelling of a material is a knowledge acquisition and representation process, in which the knowledge to be acquired and represented is the complex behaviour of a material. Clearly, the learning or self-organising capability of neural networks can thus be utilized to build a model of the behaviour of a material, given an appropriate amount of data on that material's response to external stimuli. This realisation, along with developments in hardware-based programmable neural networks and neural computing theory, have drawn a potentially new horizon for research in material modelling. The first two authors and their associates were the first who proposed to use neural networks for material modelling [1-5].

The aim of this paper is to introduce to the Lithuanian engineering and research society a relatively new approach of material modelling based on neural network method. The paper presents: 1) a description of backpropagation neural networks, 2) a brief review of higher order learning and adaptive architecture determination techniques, 3) a summary on the neural network modelling procedures, and 4) a description of the concept and principles of the neural network-based material modelling.

2. Brief description of backpropagation neural networks

2.1. General

Neural networks are computational models inspired by our understanding on the biological structure of neurons and the internal operation of the human brain. Research in neural networks was started in the 1940s when an endeavour in the search for means of constructing a brain-like computing machine was undertaken, and the mathematical foundation for this learning paradigm was essentially laid during that period. The first computational model of a neuron or a processing unit in a neural network, which is capable of threshold logical operation, was proposed by McCulloch and Pitts [6].

A neural network is a non-linear dynamic system consisting of a large number of highly interconnected processing units, or processors. Each processing unit in the network maintains only one piece of dynamic information (its current level of activation) and is capable of only a few simple computations (adding inputs, computing a new activation level, or performing threshold logical calculation). A neural network performs "computations" by propagating changes in activation between the processors; it stores the knowledge it has "learned" as strengths of the connections between its processors. The large number of these processing units, and even larger number of inter-connections, similar to the neuronal structure of human brain, give the neural networks their capability of knowledge representation. In addition, it is

through self-organisation or "learning" that a neural network approaches some representation of a particular knowledge or discovers the hidden relationships in data.

Self-organisation or "learning" is a key characteristic of neural networks. Unlike traditional sequential programming techniques, neural networks are trained with examples of the concepts to capture. The network then internally organises itself to be able to reconstruct the presented examples.

The operation of a processor in a neural network computation is very simple. The output of a processor, which is computed from its activation level and many times is the same as the activation level, is sent to other "receiving" processors via the processor's outgoing connections. Each connection from one to another processor possesses a numeric weight representing the strength or weight of the connection. The strength of connection is a filter (in the form of a multiplicative coefficient) of the output sent from one processor to another processor, and may serve to increase, or decrease, the activation of the receiving processor. Each processor computes its activation level based on the sum of the products of connection strengths and outputs coming into the processor over its incoming connections, computes its output based on this net input, and then sends its output to other processor to which it has outgoing connections.

The propagation of activation in a neural network can be feedforward, feedback, or both. In a feedforward network, a type of signal can be propagated only in a designated direction, whereas in a network with feedback mechanism this type of signal can flow in either direction or recursively. For example, in a strictly feedforward multilayer network, only inter-layer connections between adjacent layers are allowed, and the inter-layer connections or lateral connections among nodes in the same layer are suppressed.

The network topology, and the form of the rules and functions are all learning variables in a neural network learning system and lead to a wide variety of network types.

2.2. Backpropagation neural networks

Backpropagation networks and their variants, as a subset of multilayer feedforward networks, are currently the most widely used networks in applications. The backpropagation neural network is given its name due to the way that it learns – by backpropagating the errors seen at the output nodes. The major distinction among feedforward neural networks is manifested by the learning rule utilised. The backpropagation network is a multilayer feedforward neural network with the generalised delta rule as its learning rule.

The processing units in a backpropagation neural network as shown in Fig 1 are arranged in layers. Each neural network has an input layer, an output layer, and a number of hidden layers. Propagation takes place in a feed forward manner, from input layer to the output layer. The pattern of connectivity and the number of processing units in each layer may vary with some constraints. No communication is permitted between the processing units within a layer. The processing units in each layer may send their output to the processing units in higher layers.

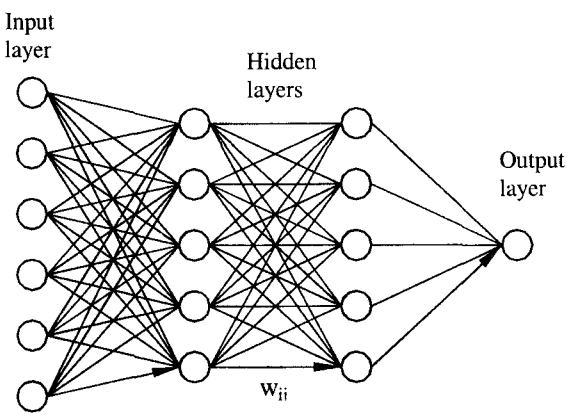

Fig 1. An example of backpropagation neural network

Associated with each connection is a numerical value which is the strength or the weight of that connection: w_{ij} = strength of connection between units i and j. The connection strengths are modified during the training of the neural network. At the beginning of a training process, the connection strengths are assigned random values. As examples are presented during the training, through application of the "rule of learning", the connection strengths are modified in an iterative process. As the successful completion of the training, when the iterative process has converged, the collection of connection strengths of the whole network has captured and stored the knowledge and the information presented in the examples used in its training. Such a trained neural network is ready to be used. When presented with an input pattern, a feed forward network computation results in an output pattern

which is the result of the generalisation and synthesis of that it has learned and stored in its connection strengths.

Therefore, in a backpropagation network, two computational procedures are performed in a learning cycle: the feedforward computation of activation and the backward propagation of error signals for the modification of connection weights via the generalised delta rule. A feedforward computation proceeds as follows:

1) The units in the input layer receive their activations in the form of an input pattern and this initiates the feed forward process;
2) The processing units in each layer receive outputs from other units and perform the following computations:
   a) Compute their net input N_j,

$$N_j = \sum_{k=1}^{M} w_{jk} o_k \,, \tag{1}$$

where o_k = output from units impinging on unit j, and M = number of units impinging on unit j.

   b) Compute their activation values from their net input values,

$$a_j = F_j(N_j), \tag{2}$$

where F_j is usually a sigmoid function and its exact form is determined by the specified range of activation values. For example, if the activation values are taken in the range of (− 1.0, 1.0), then $F(N) = 2.0(1 / (1+e^{-N-\theta}) - 0.5)$, where θ is the bias value at that processing unit.

   c) Compute their outputs from their activation values. Usually, the output is taken the same as the activation value.

$$o_j = a_j. \tag{3}$$

3) The output values are sent to other processing units along the outgoing connections.

Several mechanisms for imparting self-organisation or learning to these multilayer feedforward networks have been developed. One form of supervised learning, developed by Rumelhart et al. [7], is called the *generalised delta rule* and is the learning mechanism used in backpropagation neural networks. The modification of the strengths of the connections in the generalised delta rule, as described in [7], is accomplished through performing the gradient descent on the total error space in a given training case.

$$\Delta w_{ij} = \eta \nabla E(w_{ij}) = \eta \delta_j o_i. \tag{4}$$

In this equation, $\eta = a$ learning constant called the "learning rate", $\nabla E(w_{ij})$ = gradient of the total error with respect to the weight between units i and j, and δ_j = gradient of the total error with respect to the net input at unit j. At the output units δ_j is determined from the difference between the expected activations t_j and the computed activation a_j:

$$\delta_j = (t_j - a_j) F'(N_j) \tag{5}$$

where F' is the derivative of the activation function.

At the hidden units the expected activations are not known a priori. The following equation calculates δ_j for the hidden units:

$$\delta_j = \left(\sum_{k=1}^{M} \delta_k w_{jk} \right) F'(N_j). \tag{6}$$

In this equation, the error attributed to a hidden unit depends on the error of the units it influences. The amount of error from these units attributed to the hidden unit depends on the strength of connection from the hidden unit to those units; a hidden unit with a strong excitatory connection to a unit exhibiting error will be "blamed" for this error, causing this connection strength to be reduced.

3. Higher order learning and adaptive architecture determination

As has been stated in the previous section, the generalised delta rule [7] is basically a steepest descent scheme with constant step length in a network setting, performing a gradient descent on the error function with respect to the weight space. For multilayer feedforward neural networks, the error function is usually a highly non-linear function defined as:

$$E(w) = \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} E_k \,, \tag{7}$$

where $E_k = |t(x_k) - o(x_k, w)|^2$; $t(x_k)$ is the expected output; $o(x_k, w)$ is the network prediction which is a function of the input vector x and network weight vector w; and N is the number of training cases. This error surface is dominated with flat areas and troughs, which render the learning with the generalised delta rule in a backpropagation network very slow. Another drawback of a standard backpropagation network is the need for pre-

determination of network architecture and the inability to incorporate a priori possessed knowledge.

The modelling capability and performance of a backpropagation network is mainly determined by the network architecture and its rule of learning. Recently, several approaches have been proposed to improve the performance of backpropagation neural networks. In general, there are five ways to approach a solution to this problem: 1) using better data representation scheme for input and output, 2) employing higher order learning algorithms or heuristic algorithms that more quickly find the minimum of the error surface [8-11], 3) preprocessing the input pattern, introducing independence into the input vector space [12], thus facilitating the determination of the decision space, 4) designing innovative training schemes so that certain knowledge is pre-oriented in the network before the final training session [13,14], and 5) incorporating network geometry adaptation with efficient learning algorithms.

From the derivation of the generalised delta rule, it is tempting to postulate that all the minimisation schemes are applicable as learning rules for multilayer feedforward networks. Furthermore, numerical analysis tells us that higher order schemes such as Newton's method, quasi-Newton methods, and Conjugate Gradient methods have better numerical properties than the steepest descent method with respect to the rate of convergence and numerical stability [15,16]. Nevertheless, for neural network learning algorithms which are eventually to be employed in massively parallel hardware implementation of these networks, it is desirable that they not only be computationally efficient, but also suitable for implementation via local update only, thus conserving the parallelism of network operations. With the generalised delta rule, the formula for weight update with a momentum term is:

$$\Delta w(t) = -\eta \partial E / \partial w(t) + a \Delta w(t-1), \qquad (8)$$

where η is the learning rate and α the momentum factor, and both of them are assumed constants. The update of weights can be proceeded either in batch mode or in online mode. The former refers to updating weights after all the training sets have been presented, and the later after each training set. For second and higher order algorithms with adaptive determination of η or α, the update of weights is usually implemented in batch mode. To date, numerous new learning schemes have been proposed, such as the Quickprop algorithm [9], the Delta-Bar-Delta algorithm [10], the Pseudo-Newton algorithm [8], and quasi-Newton style methods [17], etc, using either heuristic rules or higher order information to compute the learning parameters. Experience shows that heuristic rules are simple, robust, and computationally efficient, while the acquisition of higher order information is usually computationally expensive.

Except for some trivial problems, the network architecture on the hidden layers cannot be determined in advance. The common approach to architecture determination uses trial and errors, for simple problems. For real world engineering problems such as material modelling, it is imperative to have adaptive or dynamic mechanisms to determine the network architecture. Since the input and output of a network are determined by the nature of the problem and the representation scheme selected, adaptive schemes for architecture determination have adopted mechanisms of either "growing" or "pruning" the number of processing in hidden layers. A "growing" process starts with a basic or small network (usually one or a small number of hidden units), and then adds or grows additional processing units or a set of units including layer(s) to the network as the training process progresses until the convergence of training is reached. A "pruning" process usually starts with a larger network than needed, and then deletes redundant processing units or links during or after a training session with the hope that the generalisation capability of the trained network would be improved. Sometimes, "pruning" is also performed on nodes in the input and output layers in order to determine the most important set of variables in the representation scheme. The former approach is represented in the Dynamic Node Creation scheme [18], the Cascade Correlation Learning Architecture [13], and the Self-Organising Neural Network [14], and the latter in Skeletonization [19], and Karnin's pruning scheme [20].

In general, the "growing" approach is more efficient and robust than the "pruning" scheme for the determination of network architecture. For certain classification problems, pruning can be incorporated to improve network generalisation. However, for real value functional mapping problems in which accuracy on predictions becomes more demanding, pruning might have an adverse effect.

On functional mapping, theoretical studies have proven that a multilayer feedforward network with one

hidden layer and enough hidden nodes is a universal approximator, ie, any function can be embedded in a three layer network [21, 22]. This conclusion is valid in the limit sense of statistical measurement. However, for efficiency in learning, two or more hidden layers are usually used in applications [23].

For material modelling problems investigated by the authors [2-4], the architecture adaptation scheme is based on "growing" approach [18] with the use of two hidden layers.

4. Neural network modelling procedure

In applying neural networks as a computational and knowledge representation tool to solve any non–trivial problem, the modelling process usually involves the followings aspects: 1) problem representation, 2) architecture determination, 3) learning process determination, 4) training of the neural network with training data, and 5) testing of the trained network with testing data for generalisation evaluation. These five aspects also constitute the framework of the neural network-based material modelling process to be described later.

In general, the problem representation process consists of evaluating the applicability of the neural network paradigm, the selection of the type of neural networks, data acquisition, data processing, and the design of representation schemes for input to and output from the network. The representation schemes are determined not only by the nature of the problem, but also by the way that models are to be used. There are basically two kinds of representation schemes: distributed representations and local representations. For function mapping problems such as material modelling, local representation scheme is usually adopted.

Architecture determination usually involves the selection of the number of layers and nodes in each layer, as well as the interconnection scheme. Obviously the size of the input and output layer is solely determined by the representation scheme devised. However, the size of each hidden layer and the number of hidden layers are strongly influenced by the complexity of the problem, features or regularities embedded in the training data, and the efficiency of learning algorithms. In other aspect, the way that nodes in different layers are connected is also very important because it controls the pathway for information flow or propagation in a network. Though the connection between layers can be forward, backward, and recurrent, or be established between subsets of processing units in different layers, for simplicity, complete connection between adjacent layers is usually enforced in multilayer feedforward neural networks, especially when dealing with function mapping problems.

After the data representation scheme and initial network architecture are defined, the determination of a generic learning process involves making decision on the type of processing units such as the Σ unit and the $\Sigma\Pi$ unit (where Σ is summation and Π is multiplication), the selection of activation function, and the design of learning algorithms. Once the whole learning system is constructed, the training and testing process can be performed.

Training means that the defined network is presented with processed training data and learns or discovers the relationships embedded in the data using learning algorithms. Convergence of learning is reached if the error associated with the network prediction falls within a specified error tolerance. If a presentation of the whole training data to the network is defined as a learning cycle or an epoch, the iterative training process usually requires many hundreds or thousands epochs to reach convergence. After the network is properly trained, its generalisation capability is evaluated in the testing phase. If the trained network generalises reasonably well on novel but similar cases, the resulting neural network can then be qualified as a legitimate model for use in the problem domain.

For real world engineering problems, this whole modelling process is likely to be an iterative process, and the generalisation evaluation on the trained network from the testing phase functions more like a feedback signal. Since a neural network learning system is an integration of different mutually interacting learning components, one or sometimes even all of the previous processes may need to be examined and adjusted if the generalisation capability of the trained network is unsatisfactory. The discrepancy between the expected output and network prediction may be result from any of the following sources: 1) an inappropriate representation scheme of the problem; the training data is not comprehensive enough to represent the essence of the problem; or the domain is not suitable to neural networks; 2) the current architecture of the network is insufficient to accommodate the knowledge to be captured; 3) the learning algorithm is not efficient and robust enough to handle the complexity of the problem; and 4) the training is pre-maturely terminated.

5. Neural network-based material modelling methodology

The basic strategy for developing a neural network-based model of material behaviour is to train a multilayer feedforward neural network on the stress-strain results (data) from a series of experiments on a material. If the experimental data about the material behaviour are fairly comprehensive, the trained neural network would contain sufficient information about the material behaviour to qualify as a material model. Such a trained neural network not only would be able to reproduce the experimental results it was trained on, but through its generalisation capability it should be able to approximate the results of their experiments on the same material. The degree of accuracy in this generalisation depends on both how comprehensive and representative the training set is and how well the network is trained.

Clearly, the procedures used in the construction of a neural network-based constitutive model of a material would fall into the general framework of the neural network modelling process described in the previous section. Because of the nature of a material model and its intended use within the finite element method, the modelling procedure has its own characteristics and requires special considerations.

As has been mentioned before, the first step in constructing a neural network-based material model is the determination of representation scheme for material behaviour in the input and output. The composition of the input and output layers depends primarily on the intended use of the neural networks. Although neural networks offer considerable flexibility in this regard, it is natural that the first attempt in the development of neural network-based material models should follow the traditional mathematical models for use with finite element methods. As such, the processing units in the input and output layers all represent stresses, strains, their increments, and in some cases a portion of the stress-strain history. Since the material behaviour is highly path dependent, the input must have sufficient information for the neural network to characterize the stress-strain state of the material and contain certain information on the previous history. Therefore, two representations schemes – the so-called one-point and three-point schemes, are introduced to characterize the behaviour of a material in different stress states. These representation schemes can be either stress-controlled which means that the network is to predict strain increments corresponding stress increments, or strain-controlled on the contrary.

For instance, in a stress–controlled one-point representation scheme, the stress-strain state of a material at one point in the stress space and strain space and the next stress increments at that point are included in the input, and the corresponding strain increments are in the output. For a strain–controlled one-point representation scheme, however, the strain increments are in the input and stress increments are in the output. The three-point representation scheme is an expansion of the one-point scheme, with an expanded input including two additional stress-strain states in the stress-strain history.

Decisions regarding the neural network architecture are of primary importance in the successful construction of neural network-based material models. The capacity of a neural network is a function of the number of hidden layers and the number of processing units in each layer [22]. The pattern of connectivity between the layers is also part of this equation. However, in this study a simple pattern of connectivity is used: each processing unit has outgoing connections to all the processing units in the next layer. The capacity of the neural network is also somehow related to the amount of the information in the training data and the complexity of the knowledge contained in that data. Currently there are no quantitative theories or good qualitative rules for determining the capacity of a multilayer feedforward neural network, as this aspect is not yet well understood. Though theoretical studies have concluded that one hidden layer with enough hidden nodes can accomplish the modelling of any functions [21, 22], in practice, especially with modelling of continuous functions, it has been observed that the use of two hidden layers would yield a more efficient training. According to the authors' experience, two hidden layers are used for material modelling problems. With the use of two hidden layers, the size of each hidden layer is determined by the modified dynamic node creation scheme. Consequently, the final size of each hidden layer thus determined corresponds to the network architecture when a minimal or optimal training data set is successfully trained. This minimal or optimal training data set is defined as a set of data that contains sufficient information to characterise the behaviour of a material.

Whether or not a neural network has been trained with the minimal training data set is indicated by how well the trained network generalises on the testing cases. Ideally, if the network is trained with a quasi-optimal or quasi-minimal training set, reasonable generalisation should be observed on the testing results. Otherwise, if the training set is too small, poor testing performance would be expected, as the trained network has not been presented with all examples of the relevant information so as to generalise properly. On the other hand, if the training data set is too large, no substantial improvements would result from further training, after the network has been trained with the minimal training data set.

In the incremental training scheme proposed [2], training and testing proceed in the following way: 1) start with a small network and a small training set, and train the network until convergence; 2) add additional data to the training set, and restart training on the augmented data with the previously converged network; add nodes to hidden layers; 3) when a training set containing a reasonable number of stress-strain data has been successfully trained, perform the generalisation tests on untrained stress-strain cases; and 4) if all the testing results appear in good agreement with expected behaviour, stop training; otherwise repeat the data set addition and generalisation testing processes.

There are some benefits to using incremental training with tests for generalisation evaluation. First, with the use of the dynamic node generation scheme and incremental presentation of the entire training data set, the network is not overwhelmed by the large amount of information at the initial stage of training so that the learning process converges faster than when guessing a network architecture and presenting the network with the whole training set at once. Secondly, starting with a small amount of data and monitoring the generalisation performance of the neural network at certain stages of training, a quasi-minimal training set can usually be obtained. However, the true minimal training set is not theoretically defined at this time, but it is known to depend on both the comprehensiveness of the available experimental data on a material and the characteristics of the problem.

Concluding remarks

Research interest in neural networks, as a paradigm of computational knowledge representation, has experienced considerable increase in recent years. This new interest is supported by the realisation that neural computing is inherently parallel and functionally more close to the operation of the brain; that is, it has the capability of self-organisation or learning. With the advance and sophistication in some branches of neural-networks, the technology has been successfully tailored for a wide range of problems, such as the modelling of some cognitive processes, vision, image processing, pattern recognition, and some engineering fields. It is obvious that with the continuous development on the computational theory and hardware implementation of neural networks, this technology will potentially provide an efficient and viable tool for solving certain engineering problems that are difficult for mere conventional approaches.

Research in the application of neural networks to problems in computational mechanics is quite recent [1]. It can be concluded that the use of neural networks for the modelling of material behaviour is viable and promising. Such an approach does not make a priori assumptions about the behaviour of a material, but rather bases its prediction of stress-strain behaviour on the experimental data with which it has been trained.

References

1. J. Ghaboussi, J. H. Garrett and X. Wu. Knowledge-Based Modelling of Material Behaviour Using Neural Networks // ASCE Journal of Engineering Mechanics Division, Jan. 1991.
2. X. Wu and J. Ghaboussi. Neural Network-Based Material Modelling // Technical Report No 599, Structural Research Series, Dept. of Civil Eng., University of Illinois at Urbana-Champaign, 1995. 198 p.
3. J. Ghaboussi, D. A. Pecknold, M. Zhang and R. HajAli. Neural Network Constitutive Models Determined from Structural Tests // Proceedings, 11th ASCE Engineering Mechanics Conference, Ft. Lauderdale, May 1996.
4. G. Kaklauskas, J. Ghaboussi, and X. Wu. Neural Network Modelling of Tension Stiffening Effect for R/C Flexural Members // Proceedings, EURO-C 1998-Computational Modelling of Concrete Structures, Badgastein, Austria, March 31 – April 3, 1998, p. 823–832.
5. J. Ghaboussi and D. Sidarata. A New Nested Adaptive Neural Network for Modelling of Constitutive Behaviour of Materials // International Journal of Computers and Geotechnics (to be published).
6. W. S. McCulloch and W. Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity // Bulletin of Mathematical Biophysics 5, 1943, p.115–133.
7. D. E. Rumelhart, G. E. Hinton and R. J. Williams. Learning Internal Representations by Error Propagation // Parallel Distributed Processing, Vol 1: Foundations, D.E. Rumelhart and J. McClelland (Eds.), The MIT Press, MA, 1986, p. 318–362.

8. S. Becker and Y. Le Cun. Improving the Convergence of Backpropagation Learning with Second Order Methods // Proceeding of the 1988 Connectionist Models Summer School, Cernegie Mellon University, Pittsburgh, 1988.
9. S. E. Fahlman. Faster-Learning Variations on Backpropagation: An Empirical Study // Proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufmann Publishers, Inc., 1988.
10. R. A. Jacobs. Increased Rate of Convergence through Learning Rate Adaptation // Technical Report (COINS TR 87-117). Dept. of Computer and Information Science, University of Massachusetts at Amherst, MA, 1987.
11. J. Moody. Fast Learning in Multi-resolution Hierarchies // D. S. Touretzky (Ed.). Advances in Neural Information Processing Systems 1, Morgan Kaufmann, 1989.
12. S. J. Orfanidis. Gram-Schmidt Neural Nets // Neural Computation 2, 1990, p. 116–126.
13. S. E. Fahlman and C. Lebiere. The Cascade-Correlation Learning Architecture // Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, Feb. 1990.
14. M. F. M. Tenorio and W.-T. Lee. Self-Organizing Neural Networks for Optimum Supervised Learning // TR-EE-89-30, School of Electrical Engineering, Purdue University, June 1989.
15. L. A. Hageman and D. M. Young. Applied Iterative Methods. The Academic Press, New York, 1981.
16. G. H. Golub and C. F. Van Loan. Matrix Computations. The Johns Hopkins University Press, 1983.
17. R. L. Watrous. Learning Algorithm for Connectionist Networks: Applied Gradient Methods of Nonlinear Optimization // Proceedings of the IEEE International Conference on Neural Networks, Vol II, 1987, p. 619–627.
18. T. Ash. Dynamic Node Creation in Backpropagation Networks // ICS Report 8901, Institute for Cognitive Science, University of California, San Diego, La Jolla, Feb. 1989.
19. M. C. Mozer and P. Smolensky. Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment // *CU–CS–421–89*, Dept. of Computer Science, University of Colorado at Boulder, Jan. 1989.
20. E. D. Karnin. A Simple Procedure for Pruning Backpropagation Trained Neural Networks // IEEE Transactions on Neural Networks, June 1990, Vol 1, No 2, p. 239–242.
21. G. Cybenko. Approximations by Superpositions of a Sigmoidal Function // CSRD Report No 856, Center for Supercomputing Research and Development, University of Illinois at Urbana–Champaign, Feb. 1989.
22. K. Hornik, M. Stinchcomebe and H. White. Multilayer Feedforward Networks Are Universal Approximators // Neural Networks, Vol 2, 1989, p. 359–366.
23. R. P. Lipman. An introduction to Computing with Neural Nets // IEEE ASSP Magazine, April 1987, p. 4–22.

Įteikta 1999 05 20

MEDŽIAGŲ NEURONINIŲ TINKLŲ FIZIKINIAI MODELIAI

J. Ghaboussi, X. Wu, G. Kaklauskas

Santrauka

Straipsnyje supažindinama su neuroninių tinklų metodo taikymu, kuriant fizikinius medžiagų modelius. Neuroninių tinklų metodas, pagrįstas žmogaus smegenų darbo modeliavimo principais, tik šį dešimtmetį praktiškai pradėtas taikyti įvairiose mokslo srityse. Pirmieji du šio straipsnio autoriai pirmieji pasaulyje pritaikė neuroninių tinklų metodą fizikiniams modeliams kurti.

Neuroninį tinklą sudaro mazgai (neuronai), tarpusavyje sujungti ryšiais. Mazgai yra suskirstyti į grupes, vadinamas sluoksniais: pradinių duomenų ir rezultatų sluoksniai bei tarpiniai sluoksniai (1 pav.). Mazgai charakterizuojami aktyvumu, o ryšiai stiprumu. Mazgo aktyvumas nustatomas kaip į jį ateinančių ryšių stiprumo ir atitinkamų mazgų aktyvumo sandaugų suma. Ryšių stiprumas, kuris gali turėti tiek teigiamą, tiek neigiamą skaitinę reikšmę, nustatomas neuroninio tinklo „mokymo“ metu. Tinklas dažniausiai „mokomas“ pradinių duomenų ir rezultatų pavyzdžiu pagal tam tikras mokymo taisykles. Iš visų žinomų neuroninių tinklų bene plačiausiai taikomas grįžtamasis neuroninis tinklas (backpropagation neural network).

Straipsnyje supažindinama su grįžtamuoju neuroniniu tinklu, jo „mokymo“ taisyklėmis, dinaminiais mazgų kūrimo principais bei tinklų kūrimo metodologija. Straipsnio pabaigoje pateikiama medžiagų fizikinių modelių kūrimo neuroniniais tinklais metodologija.

Jamshid GHABOUSSI. Professor. University of Illinois, 3118 Newmark Civil Engineering Laboratory 205 North Mathews Avenue Urbana, Illinois 61801, USA.

PhD (1971, civil engineering) from the University of California at Berkeley. Since 1973 he served on the faculty of University of Illinois at Urbana-Champaign. Full professor since 1984. Over 25 years of experience in research and development in several fields of engineering computation. Author of more than 100 publications. Consultant on numerous major engineering projects.

Xiping WU. Research Engineer in Offshore Division, Exxon Production Research Company in Houston, Texas.

PhD (1991, structural engineering) from the Dept of Civil Engineering, University of Illinois at Urbana-Champaign. He jointly developed the neural network-based methodology for constitutive modelling and structural damage assessment, as well as innovative neural network learning algorithms and architecture. More than 30 professional publications and 7 years of academic research and teaching experience in Singapore and USA in structural engineering, computational mechanics and engineering applications of soft computing methods. Research interests: soft computing methods, structural dynamics, earthquake engineering, structural reliability analysis and ice mechanics.

Gintaris KAKLAUSKAS. PhD, Senior Researcher and Associate Professor. Dept of Reinforced Concrete Structures, Vilnius Gediminas Technical University, Saulėtekio al. 11, 2040 Vilnius, Lithuania.

A graduate of Vilnius Civil Engineering Institute (presently Vilnius Gediminas Technical University) (1982, civil engineer). PhD (1990). Research visits: Aalborg University (Denmark, 1991), University of Glamorgan (UK, 1994/1995), University of Illinois, Urbana-Champaign (USA, 1996). Author and co-author of 2 monographs, 1 invention and a number of papers. Research interests: development of constitutive relationships for concrete and numerical simulation of reinforced concrete structures.