Structural Engineering and Mechanics, Vol. 6, No. 8 (1998) 955-969
DOI: http://dx.doi.org/10.12989/sem.1998.6.8.955

Soft computing with neural networks for engineering applications: Fundamental issues and adaptive approaches

Jamshid Ghaboussi †

Department of Civil Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801, U.S.A.

Xiping Wu ‡

Offshore Division Exxon Production Research Co., Offshore Division, P.O. Box 2189 Houston, Texas 77252-2189, U.S.A.

Abstract. Engineering problems are inherently imprecision tolerant. Biologically inspired soft computing methods are emerging as ideal tools for constructing intelligent engineering systems which employ approximate reasoning and exhibit imprecision tolerance. They also offer built-in mechanisms for dealing with uncertainty. The fundamental issues associated with engineering applications of the emerging soft computing methods are discussed, with emphasis on neural networks. A formalism for neural network representation is presented and recent developments on adaptive modeling of neural networks, specifically nested adaptive neural networks for constitutive modeling are discussed.

Key words: neural networks; soft computing; computational mechanics; constitutive models.

1. Introduction

Soft computing methods, including neural networks, genetic algorithms, fuzzy set theory, probabilistic reasoning, and integrated systems, are biologically inspired computing tools. Neural networks are inspired by the internal structure and operation of the brain; genetic algorithms are inspired by the evolutionary process and adaptation in nature; and, fuzzy logic is inspired by the natural language and linguistic approaches to problem solving. Development of soft computing methods has reached the level that they offer new opportunities in engineering applications. Recently, considerable success has been achieved in the application of these methods to diverse fields of engineering problems. In the field of mechanics, for example, neural networks and other computational intelligence tools have been used in constitutive modeling of concrete, composites and geomaterials (Ghaboussi, Garrett and Wu 1990, Wu 1991, Ghaboussi 1992, Wu and Ghaboussi 1992, 1993, Ghaboussi *et al.* 1994, 1997, Penumadu

† Professor
‡ Senior Research Engineer (Formerly, visiting Assistant Professor, Department of Civil Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois)

et al. 1994, Ellis *et al.* 1995, Ghaboussi and Sidarta 1998, Sidarta and Ghaboussi 1998, Ghaboussi *et al.* 1998), structural analysis and design (Vanluchene and Sun 1990), strain softening material models in reinforced concrete (Kaklauskus, Ghaboussi and Wu 1998), structural damage assessment (Wu, Ghaboussi and Garrett 1992, Elkordy *et al.* 1993, Ghaboussi and Banan 1994, Ghaboussi, Banan and Florom 1994, Chou and Ghaboussi 1997, 1998, Chou, Ghaboussi and Clark 1998), structural control (Chen *et al.* 1995, Ghaboussi and Joghataie, 1995, Nikzad and Ghaboussi 1997, Bani-Hani and Ghaboussi 1998, Kim and Ghaboussi 1998).

In spite of the success of soft computing methods in solving some challenging problems in engineering, because of their unique computational characteristics, there are some fundamental issues which need to be addressed. These include the issues of universality and uniqueness of the solution, imprecision tolerance and imprecise reasoning, adaptivity, redundancy and robustness, etc. A clear understanding of these issues is essential in exploiting the full potential of soft computing methods for engineering problem solving. Most of the discussion that follows is offered without rigorous proofs, which are not currently available.

2. Computational intelligence in engineering

The field of engineering deals with designing entities, building, manufacturing and maintaining them throughout their useful lives. All engineering practices are based on the scientific principles, which distinguishes the engineering profession from other trades which also design and build new objects. However, there are fundamental differences between engineering and scientific methodologies. Scientists often isolate a single phenomena or a single problem to study. In contrast, engineering takes place in the real world and it must deal with all aspects of the problem simultaneously, often in situations that lack sufficient data and, whatever data may be available is often noise contaminated and contains scatter. Necessity may dictate the use of approximate methods which may yield sufficient accuracy but may not be universally applicable. Precision in engineering methodology is relative and most engineering problems are imprecision tolerant to various degrees. This is a fundamental point. Imprecision tolerance leads to methodologies which lack universality and functional uniqueness, which in turn leads to more relaxed standards of proof. In reality, precision, universality and functional uniqueness, which are the attributes of the mathematically based methods, are less important in engineering methodology, than added tractability and robustness. Tractability is needed to add capabilities beyond those of the existing mathematically based methods and robustness is needed to deal with uncertainty and lack of sufficient information. Historically, engineering methodology has been based on mathematics and in recent decades the same methodology has been carried into the domain of computation, leading to such fields as computational mechanics. We refer to this form of engineering computing on the current serial computers as hard computing. On the other hand, soft computing methods are a class of biologically inspired methods which share the attributes of the biological problem solving methodology, such as imprecision tolerance, non-universality and functional non-uniqueness.

2.1. Hard computing methods

Nearly all the scientific and technical computing done on the current generation of

computers, from PCs to supercomputers, has to be considered as hard computing. The main characteristic of hard computing methods is that they are highly dependent on precision. Although hardware requires that the scientific and technical computations be performed in finite mathematics, the computational precision is often far greater than what is needed in the real life problem solving. Consider the boundary value problems or initial value problems in mechanics. The computational models for the boundary value problems are models such as finite element method and finite difference method. In order to perform a finite element analysis of a boundary value problem we need information on the geometry of the problem, the constitutive properties of the constituent materials, boundary conditions and the external actions such as the loads and thermal fields. These input parameters are not often known precisely, especially the constitutive properties of materials.

The hard computing methods often solve an idealized precise problem deterministically. Using the current hard computing methods usually involves three steps. First, the problem is idealized to develop the precise input data. Next, the hard computing analysis is performed within the machine precision on a sequential or parallel computer. Finally, the output to the idealized problem is obtained with much higher precision than required. This form of hard computing method is often used in evaluation of behavior and design of physical systems which are associated with certain level of uncertainty. Engineering judgement is used in an ad hoc manner to utilize the results of the hard computing analyses.

The consequence of the mathematically based precision in the hard computing methods is that there is no built-in procedure for dealing with uncertainty, lack of sufficient information, and scatter and noise in the data. All the input data must be provided and, where there is a gap, estimates must be made to provide "reasonable precise values" of the input data. Also, all the noise and scatter must be removed before providing the input data to the hard computing methods. Inherently, there are no internal mechanisms for dealing with the uncertainty, scatter and noise. Consequently, the hard computing methods generally lack robustness.

Hard computing methods are more suitable for direct or forward analysis and are often used for problems posed as direct. It is very difficult to solve the inverse problems with the current state of the hard computing methods. As will be discussed in a later section, there are many inverse problems that are either not solved currently, or a simplified version of these problems are posed and solved as direct problems.

2.2. Biologically inspired soft computing methods

Biological systems have evolved very effective and robust methods of dealing with uncertainty, lack of sufficient information, scatter and background noise. Consider the brain of human beings. It is a biological computer and it is far slower than the current computers. The signals in the human brain and nervous system travel in milliseconds, whereas, the signals in the computers are transmitted in nanoseconds. However, the human brain is capable of performing far more complex computational tasks than our computers. This is mainly attributed to the massively parallel structure and the enormous size of our brain, which by some estimates contains 10^{10} to 10^{11} neurons and 10^{13} to 10^{15} connections.

One of the striking characteristics of the biological computational tasks is that they are imprecision tolerant, similar to most real life engineering problems. The massively parallel structure of the human brain allows computations at much lower levels of precision than the hard computing tasks in our current generation of computers. However, the structure of the

brain has evolved in such a way that enormously complex computational tasks can be performed in real time and in an highly robust way, with internal mechanisms for dealing with uncertainty, lack of sufficient information and background noise. Consider the enormously difficult task of image recognition. Brains approach this problem in a fundamentally different way than our computers. They perform a distributed computation in the massively parallel structure of the brain in such a robust manner that images of objects from different angles or partially covered images are easily recognized in real time. This is one example of a large number of computational tasks wth insufficient amount of input information. There are numerous tasks that we routinely perform in our daily lives while, even the simplified versions of these tasks are extremely difficult to perform with our current generation of computers.

Neural networks are soft computing methods that are inspired by the massively parallel structure of the brains. They potentially have some of the same robustness characteristics as brains. Even though the current generation of neural networks are too small and too crude, they appear to be on the right path, and future research is likely to lead to more powerful neural network based soft computing methods. Another important feature in all biological systems that guides their survival in nature is adaptation through evolution within the ever changing environment. Genetic algorithms or evolutionary computing methods are based on the principle of adaptation and evolution in nature. In addition, fuzzy set theory provides a systematic way to deal with incomplete and imprecise sensory information linguistically. On a higher level, these soft computing tools can be integrated, even with hard computing methods, to form integrated systems to take advantage of unique capabilities of both hard and soft computing methodologies.

3. Neural networks as soft computing tools

The discussion in this section will be focused on multi-layer feed-forward neural networks which are currently the most commonly used neural networks in engineering applications. These neural networks are used to establish associations (mappings) between the vector of input variables $\boldsymbol{x}$, and the vector of output functions $\boldsymbol{y}$, within the domain of the training data set $\boldsymbol{D}=\{(\boldsymbol{x}_j, \boldsymbol{y}_j), j=1, ..., k\}$. The mathematical expression of the problem is in the form of a general interpolation function.

$$\boldsymbol{y}=\boldsymbol{f}(\boldsymbol{x}) \tag{1}$$

The parameters of the interpolation function $\boldsymbol{f}$ can be determined to satisfy the following condition.

$$\boldsymbol{y}_j=\boldsymbol{f}(\boldsymbol{x}_j);\ (\boldsymbol{x}_j, \boldsymbol{y}_j) \in \boldsymbol{D} \tag{2}$$

An example of an approximate approach to this problem is regression analysis, which uses a best fit, often posed as a minimization problem and expressed as follows. A function $\hat{\boldsymbol{f}}$ is selected,

$$\hat{\boldsymbol{y}}=\hat{\boldsymbol{f}}(\boldsymbol{x}) \tag{3}$$

and the parameters of the function are determined by minimizing the error for the training set.

$$\text{minimize } \| y_j - \hat{f}(x_j) \| \text{ for all } (x_j, y_j) \in D \tag{4}$$

In earlier publications, Ghaboussi and his co-workers have introduced a new notation to represent the feed-forward neural networks. This notation is intended to facilitate the discussion of the neural networks as well as to facilitate their use in computational mechanics. The general form of the notation is as follows.

$$F=F_{NN}\ (\{\text{input parameters}\} : \{NN \text{ architecture}\}) \tag{5}$$

Note that the second argument field is the network architecture which contains the number of nodes in each layer and some information concerning the training process. This information is an integral part of the neural network and therefore it is included in the function description. For the function in Eq. (1) the neural network representation is given by the following equation, where the network architecture field is left blank, signifying a generic neural network.

$$y=y_{NN}\ (x:) \tag{6}$$

Such a neural network is trained with the training set, such that within the domain of the training data it approximately represents the training data set. The approximation in neural networks is represented by the error vectors e_j within the domain of the training data.

$$e_j=y_j - y_{NN}\ (x_j :);\ (x_j, y_j) \in D \tag{7}$$

A neural network learns to satisfy its training data set approximately. It differs fundamentally from a mathematical interpolation function, which matches the training data set exactly. Neural networks are also different than regression analysis, which requires a specified function whose parameters are determined.

In general, the output errors for the training data depend on a number of factors relating to the complexity of the underlying process represented in the training data set, as well as the network architecture and the training process. It is important to note that it is not desirable to reduce the error too much. In the limit, if the output errors are reduced to zero, then the neural network would be functioning similar to an interpolation function and it will lose its generalization capability. The generalization capability of neural networks is the direct consequence of the imprecision tolerance.

4. Functional uniqueness and redundancy in soft computing

The question of uniqueness is of central importance in mathematical approaches to modeling of physical phenomena, finding the best solution (optimization), obtaining solutions to mathematical models of the physical phenomena, inverse problems and system identification. In short, uniqueness considerations are at the heart of the hard computing approaches which are based on mathematical modeling. Functional uniqueness does not play any role in the neuro-biological approaches to problem solving. The task that a neural network is trained to perform, to a great extent, is independent of the internal structure of the neural network. Neural networks with different internal architectures can learn to perform the same task with comparable levels of precision. In the case of brains, it is thought that the function of human brain is largely independent of the exact topology of the connections between neurons. Human brains are estimated to have 10^{13} to 10^{15} connections. It is also known that our DNAs do not have the capacity to store the data on the precise topology of

the connections.

It is important to note that imprecision tolerance plays a central role in the functional non-uniqueness in neuro-biological soft computing methods. Another consequence of the functional non-uniqueness is redundancy and redundant capacity in neural networks. Generally, redundancy plays an important role in the adaptivity of soft computing tools. We define adaptivity as the capability of accommodating new information and new data. In the case of neural networks, this translates into ability to learn new training cases, without having to start training a new neural network.

A related question deals with unique determinism, which relates to the sufficiency of data for uniquely determining a process. When we wish to establish relationships between various variables, we must first ascertain that such a unique relationship does in fact exist. In a mathematical modeling approach, we attempt to seek unique solutions and when there is insufficient data so that the uniqueness cannot be guaranteed the traditional methods fail to find an acceptable solution. Uniqueness is a mathematical concept and it has a precise meaning. When possible, it is desirable to meet the strict mathematical requirement of uniqueness. However, this is often not possible, especially for practical problems. There is a relationship between the amount of information available and the possibility of being able to find a unique solution. In general, the more information we have, the less restricted is the solution space and more likely that we can find unique solutions. However, in practical problems, it is less likely that we will have all the information needed to make a decision.

In practical engineering problems, very often, we have to make design decisions in absence of a sufficient amount of information. Biological systems often operate and make decisions with an insufficient amount of data. In our daily lives we constantly make decisions when little information is available. The underlying process, in the decision making in absence of sufficient data, seems to be a relaxation of the strict mathematical requirement of uniqueness. The biological systems seem to be able to find good solutions, not necessarily mathematically unique, in absence of sufficient data. An important requirement of the biological decision making is that they have to be in real time and a good acceptable solution is all that is needed and mathematical precision is not a necessary condition.

5. Inverse problems in engineering

Most engineering problems are inherently inverse problems. However, often we solve these problems only if they can be formulated as a direct problem, even in a much more simplified version. Nearly all the computer simulations with hard computing methods, including finite element analyses are direct problems, where a physical phenomenon is modeled and the response of the system is computed. A classical definition of inverse problems is determination of the input from the output of a system or determination of the system characteristics from the measured input and output of the system (system identification). We will also include design in the category of inverse problems. Design is an inverse problem, in that, it seeks a solution from the solution space which satisfies design specification, including esthetics.

An important characteristic of the inverse problems is that they often do not have a mathematically unique solutions. The measured response of the system may not contain sufficient information to uniquely determine the input to the system. Similarly, the measured

input and the measured response of the system may not contain enough information to uniquely identify the system characteristics. In short, there may be many solutions which satisfy the problem. This again is the problem of unique determinism in the domain of exact mathematics.

Biological systems, on the other hand, have developed highly robust methods for solving inverse problems. In fact, most of the computational problems in biological systems are in the form of inverse problems. Their basic strategy is to use approximate and imprecision tolerant learning within a domain of interest, thus forgoing the universality requirement of mathematically based approaches.

Researchers in neural networks know that a set of training data, which has been used to train a neural network, can also be used to train an inverse neural network. In this case, we are justified to call the first neural network a direct neural network. Then, the input of the inverse neural network is the same as the output of the direct neural network and the output of the inverse neural network is the input of the direct neural network. These neural networks can be shown in the following equations

$$y=y_{NN}(x:) \tag{8}$$

$$x=x_{NN}(y:) \tag{9}$$

If the data has been generated by measuring the stimulus response of a physical process, then the inverse neural network may be more difficult to train.

When unique inverse relationship exists, then both neural networks and mathematically based methods can be used to model the inverse mapping. However, when the inverse mapping is not unique, then modeling with mathematically based methods become increasing difficult, if not impossible. On the other hand, neural network based methods can deal with non-unique inverse problems by using the learning capabilities of the neural networks. In fact this is how the biological systems solve inverse problem, through learning.

Modern biologically inspired soft computing methods seem to display characteristics suitable for operating in the domains where mathematical certainty is not necessary. These methods have robust capabilities to formulate the problem in a way that reasonable and admissible solutions can be obtained even when we can show that mathematically unique relationships do not exist. An example, which is discussed in another paper (Ghaboussi and Lin 1997), is the problem of generating time histories of earthquake ground accelerations from the response spectrum. It is known that the response spectra can be uniquely determined from the time histories of ground acceleration. However, reverse relationship is not unique and time histories compatible with a specific response spectrum cannot be determined uniquely. This is a characteristic which is present in many inverse problems. The way we pose many physical problems in mathematical formulation makes the forward relationship unique and the inverse relationship non-unique. In the case of the inverse problem of finding the time history of ground acceleration from a response spectrum, it is shown that a neural network based method works and finds reasonable time histories for a given response spectrum.

The key to the ability of biological systems to find reasonable solutions to the inverse problems lie in learning and being able to generalize what has been learned. Unlike mathematical proofs and strict uniqueness concepts, which are universal requirements, and they are valid over all the possible ranges of variables, the soft computing solutions to inverse

problems are only valid over restricted ranges of these variables. There is a practical imperative for this requirement which applies equally to biological system decision making as well as decision making in engineering design. Mathematically exact and unique solutions are not required. The difference between mathematically exact and unique solutions and a good admissible solution is of little consequence for the outcome of the decision making process. Soft computing techniques seem to display the same characteristics which makes them useful tools for finding good admissible solutions within a restricted space of variables and parameters. The restriction in the range of variables and parameters is also based on a practical imperative. The universality of the mathematically precise methods are of little value in most practical problem solving.

6. Adaptation and adaptive modeling with neural networks

In the remainder of this paper, we will discuss the issue of adaptivity and adaptive modeling with neural networks, specifically the newly developed nested adaptive neural networks in constitutive modeling of engineering materials in computational mechanics.

Adaptation is an essential operator in the evolution process for all the biological systems. On a micro-scale level, adaptation can be considered as dealing with progressive modification of some structures by some modifiers or operators (Holland 1975). For example, with multi-layer feedforward neural networks, which are most suitable for developing applications with nonlinear function mapping type problems, the architecture of a network is determined by the pattern of connections and the connection weights. The objective of an adaptation process is to produce structures that perform on an optimal level in a given environment. With neural networks, the performance is measured by its learning capabilities and efficiency. At the initial stage, the adaptive process does not have enough information about the relative fitness of various structures. To reduce this uncertainty, the process must test the performance of different structures in the environment. This concept of adaptive process provides the basic idea in adaptive determination or dynamic evolution of neural network architecture. On a higher level, especially with constitutive modeling, this evolution of structures can also be extended to the evolution of material models. For the simplest case, we will discuss the adaptive architecture determination of multi-layer feedforward neural networks. In this case, the adaptation process is incremental.

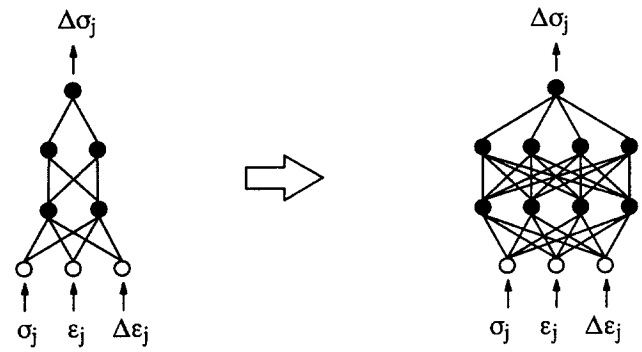
6.1. Representation and adaptive determination of multi-layer feedforward networks

The following equation describes a one dimensional neural network material model, shown in Fig. 1(a).

$$\Delta\sigma = \Delta\sigma_{NN}\ (\Delta\varepsilon,\ \sigma_j,\ \varepsilon_j:\ 3|2|2|1) \tag{10}$$

This is the so called strain-controlled one-point scheme material model. It shows that the neural network has four layers in which the input layer has three nodes representing the current stress, strain and strain increment; the output layer has one node representing the corresponding stress increment; and there are two hidden layers with two nodes in each layer.

For multi-layer feedforward neural networks, the adaptive determination of architecture is conducted on incremental determination of the number of nodes in the hidden layers during

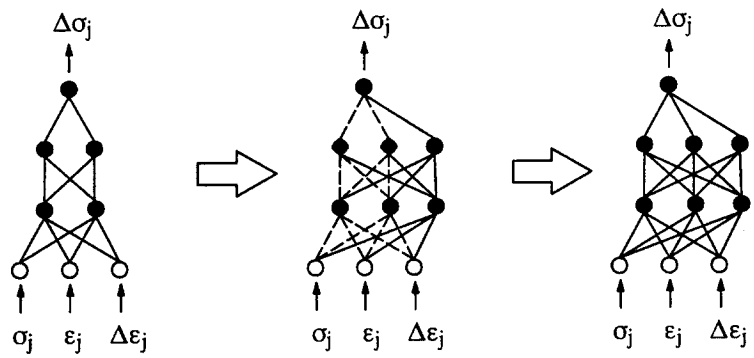


(a) Initial neural network at the start of training

(b) Final neural network at the end of training

Fig. 1 A typical neural network material model and the adaptive determination of its hidden layers during training

the training process. The training starts with an arbitrary, but small, number of nodes in the hidden layers. The learning rate is monitored during training, and as the network approaches its capacity, new nodes are added to the hidden layers, and new connections are generated. Th objective of the continued training, immediately after the addition of new nodes, is for the new connection weights to acquire that portion of the knowledge base which was not stored in the old connection weights. To achieve this, some training is done with only the new connection weights being modified, while the old connection weights are frozen. This is followed by additional cycles of training where all the connection weights are allowed to change. This process is illustrated in Fig. 2. These steps are repeated, and new nodes are periodically added to the hidden layers as needed. At the end of training, the appropriate neural network architecture has been determined automatically. The details of this algorithm is



Monitor the learning, determine when the capacity is reached and new nodes are needed

Add new nodes to the hidden layers, freeze the old connection weights, train the new connections

Unfreeze the old connections, continue training of all the connection weights

Fig. 2 The procedure for the adaptive evolution of the neural network architecture during training of the neural network material model

described in Wu (1991) Joghataie, Ghaboussi and Wu (1995). For the one dimensional neural network constitutive model shown in Fig. 1, the final neural network, after the adaptive determination of its architecture is shown in Fig. 1(b) and described by the following equation.

$$\Delta\sigma = \Delta\sigma_{NN}\ (\Delta\varepsilon,\ \sigma_j,\ \varepsilon_j:\ 3\,|\,2\text{–}4\,|\,2\text{–}4\,|\,1) \tag{11}$$

This shows that the number of hidden nodes in each hidden layer has been increased from two to four at the end of training.

6.2. Nested adaptive neural networks (NANNs)

The adaptive evolution concept of neural network architecture can be directly extended for building neural network material models. This is because most engineering data have an inherent structure and the inherent internal structure in material testing data is modular and nested (Ghaboussi and Sidarta 1998, Sidarta and Ghaboussi 1998). One type of nested structure in the material data concerns the dimensionality. This basically refers to the observation that one dimensional constitutive behavior is a subset of the constitutive behavior in two-dimensional plane strain problems, which in turn is a subset of the constitutive behavior in axisymmetric problems, which in turn is a subset of the constitutive behavior in three-dimensional state of stress.

Another form of nested structure in the material data arises from the inherent path dependency of the material behavior. From the material testing data alone, it is difficult to determine the degree of path dependency of the material behavior. For highly nonlinear behavior of materials, such as concrete under uniaxial cyclic compression, the behavior captured in a neural network material model, as expressed in Eq. (11), can only be approximate. In the past, we have included history points along the stress path in the input of the neural network material model (Ghaboussi *et al.* 1990, 1991). However, the number of history points needed in order to capture the cyclic material behavior can not be determined in advance.

With the use of adaptive evolution of network architecture, a new neural network architecture, the nested adaptive neural networks (NANN) is introduced to take advantage of nested structure in material data (Ghaboussi and Sidarta 1998). A nested neural network consists of several modules. The starting point of building a nested adaptive neural network is to develop and train a base module to represent the material behavior in the lowest function space in the data structure. The base module is a standard multi-layer feedforward neural network. However, it may be trained adaptively so that the number of nodes in the hidden layers are determined automatically during the training. The base module is successively augmented by attaching added modules to form higher level NANNs. A first level NANN is composed of the base module and a first level added module; the second level NANN is composed of a first level module and a second level added module, and so on. The process, as illustrated in Fig. 3, is theoretically open ended and more modules can be added. The added modules themselves are also standard multi-layer feedforward neural networks. In attaching a new added module to a lower level NANN, only one way connections are used; all the nodes in each layer of the new added module are connected to all nodes in the next layer of the lower level NANN. There is no connection from the lower level NANN to the new added module. The reason for the one way connections used in NANNs becomes clear if

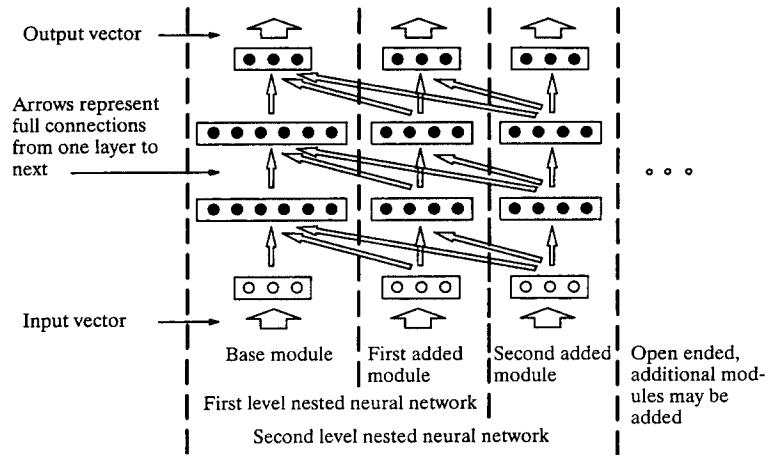
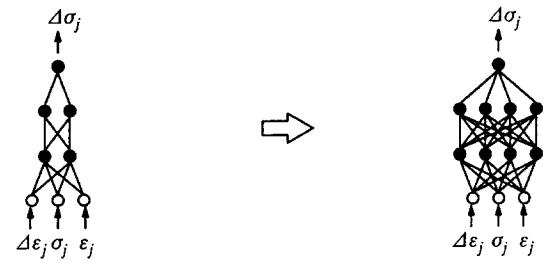
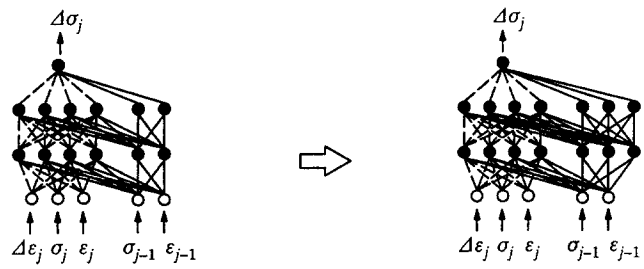


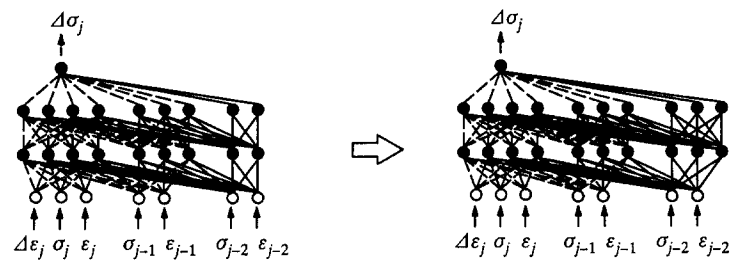
Fig. 3 Symbolic representation of a typical nested adaptive neural network



(a) The base module and its adaptive training.

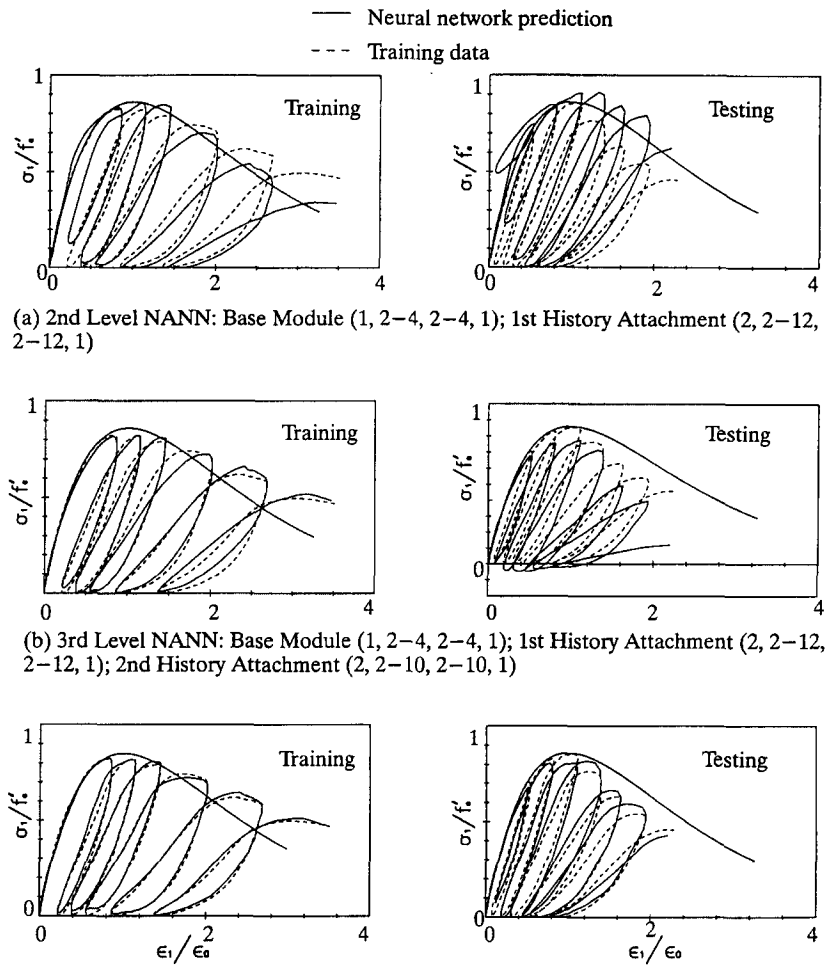


(b) Addition and adaptive training of the first history point module. The connections shown in dashed line remain frozen and only the new connection weights (solid lines) are trained.



(c) Addition and adaptive training of the second history point module. The connections shown in dashed line remain frozen and only the new connection weights (solid lines) are trained.

Fig. 4 The evolution and training of a typical nested adaptive neural network material model with two history point modules



(a) 2nd Level NANN: Base Module (1, 2−4, 2−4, 1); 1st History Attachment (2, 2−12, 2−12, 1)

(b) 3rd Level NANN: Base Module (1, 2−4, 2−4, 1); 1st History Attachment (2, 2−12, 2−12, 1); 2nd History Attachment (2, 2−10, 2−10, 1)

(c) 4th Level NANN: Base Module (1, 2−4, 2−4, 1); 1st History Attachment (2, 2−12, 2−12, 1); 2nd History Attachment (2, 2−10, 2−10, 1); 3rd History Attachment (2, 2−9 2−9, 1)

Fig. 5 Training and testing results of NANN models of concrete under uniaxial monotonic loading and cyclic compression

we consider that a new added module should not have any effect on the function space represented by the lower level module. A consequence of this observation is that the lower level NANNs should be retrievable from the higher level NANNs. To satisfy this condition, each level NANN is trained up to a satisfactory level of accuracy. After a new module is attached all the connections of the lower level NANN are frozen and only the weights of the new connections are trained. The process of development of a nested adaptive neural network model for a one dimensional constitutive model is illustrated in Fig. 4. This Figure illustrates the nested adaptive process of adding history point modules until the neural network has adequately learned the path dependent behavior of the material.

The use of NANN to adaptively determine the neural network material model has been illustrated in modeling the behavior of concrete under monotonic loading and cyclic

compression (Zhang 1996). It has been shown that the material model, specifically the number of history points, can be adaptively determined as the training and testing processes are conducted. It also identifies that four history points are required in order to fully capture the nonlinear history dependent behavior of concrete. Some typical training and testing results are shown in Fig. 5.

7. Conclusions

Soft computing methods offer new opportunities in engineering applications. Compared with hard computing methods, biological inspired soft computing methods have evolved very effective and robust ways of dealing with uncertainty, lack of sufficient data and background noise. In this paper, some fundamental issues associated with soft computing methods in engineering problem solving, such as the universality and uniqueness of solution, imprecision tolerance, approximate reasoning and adaptation are discussed. The concept of adaptation is used as the basis for adaptive determination of neural network architecture, as well as the adaptive determination of material models. The nested adaptive neural networks are introduced by using incremental adaptive procedure to take advantage of nested structures in material data.

It is reasoned that modern biologically inspired soft computing methods seem to display characteristics suitable for operating in the domain where mathematical certainty is not necessary. Because of learning and generalization capabilities of soft computing methods, reasonable and admissible solutions can be obtained even when mathematically unique relationships do not exist.

References

Bani-Hani, K. and Ghaboussi, J. (1988), "Nonlinear structural control using neural networks", *Journal of Engineering Mechanics Division, ASCE,* **124**(3),

Chen, H.M., Tasi, K.H., Qi, G.Z. and Yang, J.C.S. (1995), "Neural networks for structural control", *Journal of Computing in Civil Engineering, ASCE,* **9**(2), 168-176.

Chou, J.-H. and Ghaboussi, J. (1997), "Structural damage detection and identification using genetic algorithm", *Proceedings, International Conference on Artificial Neural Networks in Engineering,* St. Louis, Mo., Nov.

Chou, J.-H. and Ghaboussi, J. (1998), "Studies in bridge damage detection using genetic algorithm", *Proceedings, Sixth East Asia-Pacific Conference on Structural Engineering and Construction* (EASEC6), Taiwan, Jan.

Chou, J.-H., Ghaboussi, J. and Clark, R. (1998), "Application of neural networks to the inspection of railroad rail", *Proceedings, Twenty-Fifth Annual Conference on Review of Progress in Quantitative Nondestructive Evaluation,* Snowbird Utah, July.

Elkordy, M.F., Chang, K.C. and Lee, G.C. (1993), "Neural network trained by analytically simulated damage states", *Journal of Computing in Civil Engineering, ASCE,* **7**(2), 130-145.

Ellis, G.W., Yao, C., Zhao, R. and Penumadu, D. (1995), "Stress-strain modeling of sands using artificial neural networks", *Journal of Geotechnical Engineering, ASCE,* **121**(5), 429-435.

Ghaboussi, J. (1992), "Potential applications of neuro-biological computational models in geotechnical engineering", *Proceedings 4th International Symposium on Numerical Models in Geomechanics,* Swansea, U.K.

Ghaboussi, J. and Banan, M.R. (1994), "Neural networks in engineering diagnostics", *Proceedings, Earthmoving Conference*, Society of Automotive Engineers, Peoria, Illinois, SAE Technical Paper No. 941116.

Ghaboussi, J, Banan, M.R. and Florom, R.L. (1994), "Application of neural networks in acoustic wayside fault detection in railway engineering", *Proceedings, World Congress on Railway Research*, Paris France.

Ghaboussi, J., Garrett, Jr., J.H. and Wu, X. (1990), "Material modeling with neural networks", *Proceedings, International Conference on Numerical Methods in Engineering: Theory and Applications*, Swansea, U.K.

Ghaboussi, J., Garrett, Jr., J.H. and Wu, X. (1991), "Knowledge-based modeling of material behavior with neural networks", *Journal of Engineering Mechanics, ASCE,* **117**(1), 132-153.

Ghaboussi, J. and Joghataie, A. (1995), "Active control of structures using neural networks", *Journal of Engineering Mechanics, ASCE,* **121**(4), 555-567.

Ghaboussi, J., Lade, P.V. and Sidarta, D. (1994), "Neural network based modeling in geomechanics", *Proceedings, Eighth International Conference on Computer Methods and Advances in Geomechanics,* Morgantown, West Virginia.

Ghaboussi, J, Zhang, M., Wu, X. and Pecknold, D. (1997), "Nested adaptive neural networks: a new architecture", *Proceedings, International Conference on Artificial Neural Networks in Engineering, ANNIE'97*, St. Louis, Missouri.

Ghaboussi, J., Pecknold, D.A. and Zhang, M.-F. (1998), "Autoprogressive training of neural networks in complex system", *Proceedings, International Conference on Artificial Neural Networks in Engineering, ANNIE'98*, St. Louis, Missouri.

Ghaboussi, J. and Lin, C.-C. (1998), "New method of generating spectrum compatible accelerogram using neural networks", *International Journal for Earthquake Engineering and Structural Dynamics,* **27**, 377-396.

Ghaboussi, J. and Sidarta, D.E. (1998), "New nested adaptive neural networks (NANN) for constitutive modeling", *International Journal of Computer and Geotechnics,* **22**(1), 29-52.

Ghaboussi, J., Pecknold, D.A., Zhang, M.-F. and HajAli, R.M. (1998), "Autoprogressive training of neural networks constitutive models", *International Journal for Numerical Methods in Engineering,* **42**, 105-126.

Hajela, P. and Berke, L. (1991), "Neurobiological computational models in structural analysis and design", *Computers and Structures,* **41**(4), 657-667.

Holland, J.H. (1975), *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI.

Joghataie, A., Ghaboussi, J. and Wu, X. (1995), "Learning and architecture determination through automatic node generation", *Proceedings, International Conference on Artificial Neural Networks in Engineering, ANNIE95*, St. Louis, Missouri.

Kim, Y.-J. and Ghaboussi, J. (1998), "A new method of reduced order feedback control using genetic algorithm", *International Journal for Earthquake Engineering and Structural Dynamics.* (to appear)

Nikzad, J., Ghaboussi, J. and Paul, S.L. (1996), "A study of actuator dynamics and delay compensation using a neuro-controller", *Journal of Engineering Mechanics Division, ASCE,* **122**, 966-975.

Kaklauskas, G., Ghaboussi, J. and Wu, X. (1998), "Neural network modeling of tension stiffening effects for RC flexural members", *Proceedings, European Conference on Reinforced Concrete*, Vienna, Austria, June.

Penumadu, D., Jin-Nan, L., Chameau, J.-L. and Sandarajah, A. (1994), "Anisotropic rate dependent behavior of clays using neural networks", *Proceedings, XIII ICSMFE*, **4**, 1445-1448, New Delhi, India.

Shrestha, S.M. and Ghaboussi, J. (1997), "Genetic algorithm in structural shape design and optimization", *Proceedings, 7th International Conference on Computing in Civil and Building Engineering* (ICCCBE-VII), Seoul, South Korea.

Shrestha, S.M. and Ghaboussi, J., (1998), "Evolution of optimal structural shapes using genetic algorithm", *Journal of Structural Engineering, ASCE*, **124**(8).

Sidarta, D.E. and Ghaboussi, J. (1998), "Constitutive modeling of geomaterials from non-uniform

material tests", *International Journal of Computer and Geotechnics,* **22**(1), 53-71.
Vanluchene, D. and Sun, R. (1990), "Neural networks in structural engineering", *Microcomputers in Civil Engineering,* **5**(3), 207-215.
Wu, X. (1991), "Neural network based material modeling", Ph.D. thesis, University of Illinois at Urbana-Champaign, Urbana, IL 61801.
Wu, X., Gaboussi, J. and Garrett, Jr. J.H. (1992), "Use of neural networks in detection of structural damage", *J. Computers and Structures,* **42**(4), 649-659.
Zhang, M. (1996), "Determination of neural network material models from structural tests", Ph.D. thesis, University of Illinois at Urbana-Champaign, Urbana, IL 61801.