Journal of:
Numerical Methods in Civil Engineering

Numerical Methods in Civil Engineering

# Conjugate gradient neural network in prediction of clay behavior and parameters sensitivities

**Hamed Memarian fard***

*ARTICLE INFO*

*Article history:*
*Received:*
*May 2014.*
*Revised:*
*August 2014.*
*Accepted:*
*September 2014.*

*Keywords:*
*clay behavior, back propagation, conjugate gradient, artificial neural networks*

***Abstract:***

*The use of artificial neural networks has increased in many areas of engineering. In particular, this method has been applied to many geotechnical engineering problems and demonstrated some degree of success. A review of the literature reveals that it has been used successfully in modeling soil behavior, site characterization, earth retaining structures, settlement of structures, slope stability, design of tunnels and underground openings, liquefaction, soil permeability and hydraulic conductivity, soil compaction, soil swelling and classification of soils. The method of conjugate gradients provides a very effective way to optimize large, deterministic systems by gradient descent. In its standard form, however, it is not amenable to stochastic approximation of the gradient. Here we explore a number of ways to adopt ideas from conjugate gradient and Back Propagation in the stochastic setting, using fast Hessian-vector products to obtain curvature information effectively. In our benchmark experiments the resulting highly scalable algorithms converge about an order of magnitude faster than ordinary stochastic gradient descent.*

*The objective of this paper is to provide a general view to describe this method in predicting mechanical behavior and constitutive modeling issues in geo-mechanical behavior of cohesive soil to be used in geo-mechanics. In this research the Batching Back Propagation method (BBP) has been employed and the characterized parameters are introduced as initial void ratio, liquid limit, plasticity index, natural density, moisture percent, solid density of grain, over consolidation ratio, and pre-consolidation pressure. The paper also intends to present how much the input memory may cover the accuracy of predicted behavior of standard triaxial drained and undrained tests. The paper also discusses the strengths and limitations of the proposed method compared to the other modeling approaches. Also, the sensitivity of intended parameters is investigated.*

## 1. Introduction

The engineering properties of soil and rock exhibit varied and uncertain behavior due to the complex and imprecise physical processes associated with the formation of these materials (Jaksa 1995)[1]. This is in contrast to most other civil engineering materials, such as steel, concrete and timber, which exhibit far greater homogeneity and isotropy. To compromise with the complexity of geo-mechanical behavior, and the spatial variability of these materials, traditional forms of engineering design models are justifiably simplified. An alternative approach, which has been shown to have some degree of success, is based on the data alone to determine the structure and parameters of the model. The technique is known as artificial neural networks and is well suited to model complex problems where the relationship between the model variables is unknown (Hubick 1992)[2]. It is hoped that the proposed method may attract more geotechnical engineers to pay better attention to this promising tool.

For the optimization of large, differentiable systems such as stress-strain relation of materials, methods that require the inversion of a curvature matrix (Bhagat, P.M. (2005)[3], Marquardt (1963)[4], or the storage of an iterative

** Corresponding Author: MSc. Student in Department of Solid Mechanics, Moscow State University of Civil Engineering (MGSU), Moscow, Russia, email:hmemarianfard@yahoo.com*

approximation of that inverse are prohibitively expensive. Conjugate gradient techniques, which are capable of exactly minimizing a d-dimensional unconstrained quadratic problem in d iterations without requiring explicit knowledge of the curvature matrix, have become the method of choice for such cases.

Their rapid convergence, however, breaks down when the function to be optimized is noisy, as occurs in online (stochastic) gradient descent problems. Here the state of the art is simple gradient descent, coupled with adaptation of local step size parameters. The most advanced of these methods, SMD, (Schraudolph, 1999, 2002)[5, 6], uses curvature matrix-vector products that can be obtained efficiently and automatically, (Santiago, R.A., G. Lendaris, 2005)[7]. Here we use the same curvature matrix-vector products to adopt some ideas from conjugate gradient in the stochastic setting.

The paper starts with a brief overview of the structure and operation of the Feed forward neural networks and gives a general overview of this method. Finally, the paper discusses the relative success of the proposed method in predicting clay mechanical properties and behavior.

## 2. Feed Forward Networks Concept

A basic component of many neural nets, both natural and artificial, is the feed forward network. A basic such network has the structure depicted in the following diagram :

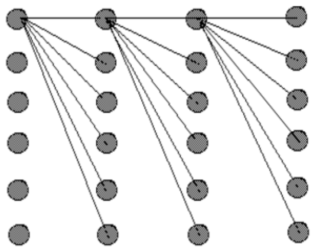


**Fig.1:** One set of connection in feed forward network

Here a layer is the usual term for a vertical row of neurons. There is full connectedness between the *n*th and *(n+1)*th layer, i.e., every neuron in the *n*th layer has a connection feeding forward into every neuron in the *(n+1)*th layer. These are not all shown in Figure 1 for reasons of clarity. Thus neurons in each layer influence neurons in the successive layer as shown in Figure 2.

The first layer contains the "input", i.e., we control activations of its neurons. For example, the first layer might represent the "retina" of a visual system, which obtains information which will be fed forward into and

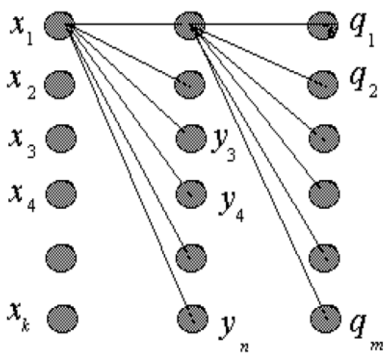


**Fig.2**

processed in further layers. The last layer contains "output", i.e., its activations provide a desired output that the neural network provides in response to input in first layer .

Funahashi (1989)[8], among others, have shown that if we desire a network which is able to take an arbitrary input pattern in the first layer, and provide an arbitrary desired output pattern in the last layer, all that is necessary is 3 layers :

More specifically, general i-o functions can be approximated in a general class of error norms using networks of this type .Henceforth we consider only 3 layer networks. We define $x_i$ to be the activation level ( either chemical or geo-mechanical potential) of the *i*th neuron in first layer, $y_i$ to be the activation level of the corresponding neuron in second layer, and $q_i$ to be the corresponding activation level in the third layer. In addition, we define $v_{ij}$ to be the strength of the connection (weight) from the *j*th neuron in layer 1 to *i*th neuron in layer 2, $w_{ij}$ to be the weight from the *j*th neuron in layer 2 to *i*th in layer 3 .

As an example, the first layer might be the retina and $x_i$ might be proportional to the illumination level at the neuron labeled $x_i$. This is the input layer - in this case light shineson retina and activates it. The last layer might represent what we would call a speech center (neuron ultimately connected to a vocal device), and its pattern $q_i$ of neuron activations corresponds to verbal description about to be delivered of what is seen in first layer .

## 3. Neuron Interaction Rule

Neurons in one layer will be assumed to influence those in next layer in almost a linear way :

$$y_i = H(\textstyle\sum_{j=1}^{k} v_{ij}\, x_j - \theta_i) \qquad (1)$$

i.e., activation $y_i$ is a linear function of activations $x_j$ in previous layer, aside from the modulating function *H*; here $\theta_i$ is a constant for each *i* .

The function *H* has on biological grounds traditionally been assumed a sigmoid as shown in Figure 3.

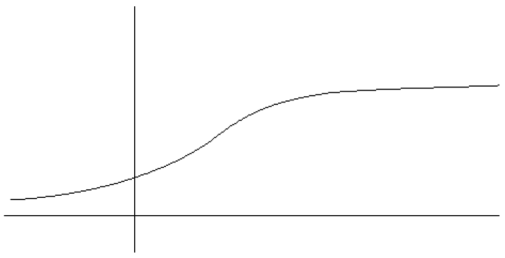


**Fig.3:** sigmoid function *H*

The function *H* has a finite upper bound, so that response cannot exceed some fixed constant .

The activation in third layer has the form of $q_i = \sum_{j=1}^{n} w_{ij} x_j$ , which is a linear function of the $y_i$ 's.

Our initial goal is to show here that we can get an arbitrary desired output pattern $q_i$ of activations on last layer as a function of inputs $x_i$ in the first layer. *x* is a vector of neuron activations in layer as follows:

$$x = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_k \end{Bmatrix} \tag{2}$$

*V* is also, a vector of connection weights from the neurons in first layer to the $i^{th}$ neuron in the second layer as follows:

$$V^i = \begin{Bmatrix} v_{i1} \\ v_{i2} \\ v_{i3} \\ \vdots \\ v_{ik} \end{Bmatrix} \tag{3}$$

Now the activation $y_i$ of the second layer is written as follows:

$$y_i = H\left(\sum_{j=1}^{k} v_{ij} x_j - \theta_i\right) = H(V^i . x - \theta_i) \tag{4}$$

The activation $q_i$ in the third layer is as follows:

$$q_i = \sum_{j=1}^{n} w_{ij} x_j = W^i . y \tag{5}$$

Therefore, the activation pattern *q* and on the last layer (output) can be made an arbitrary function of the input activation pattern as follows:

$$q = \begin{Bmatrix} q_1 \\ q_2 \\ q_3 \\ \vdots \\ q_k \end{Bmatrix} \text{ and } x = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_k \end{Bmatrix} \tag{6}$$

Note the activation of neuron in layer 3 is :

$$q_i = \sum_{j=1}^{n} w_{ij}\, y_j = \sum_{j=1}^{n} w_{ij}\, H(V^i . x - \theta_i) \tag{7}$$

Now, the question is: if $q = f\ x$ is defined by the above equation (i.e., input determines output through a neural network equation), is it possible to approximate any function in this form ?

However, if the first layer represents the retina, then if any input-output (i-o) function can be approximately encoded in the form the above equation, then it is required that if *x* represents the visual image of a chair (vector of pixel intensities corresponding to chair), then *q* represent the neural pattern of intensities corresponding to articulation of the words “this is a chair “.

Mathematically, for any given function *f(x)*: $\Re^k \rightarrow \Re$ , which it can be approximated by *f(x)*with arbitrary precision through a function $\bar{f}$*(x)*form (the above equation), presenting various measures of error, or norms (here $\Re^k$ , while *k* is a *numerator*). Then the important norms might be interested as follows :

$$\left\| f - \bar{f} \right\|_2 = \sup_{x \in \Re^k} \left| f(x) - \bar{f}(x) \right| \tag{8}$$

$$\| f - \bar{f} \|_2 = \sqrt{\int dx |f(x) - \bar{f}(x)|^2} \tag{9}$$

$$\| f - \bar{f} \|_1 = \int dx |f(x) - \bar{f}(x)| \tag{10}$$

In general form, for a certain value of $p \geq 1, \| f - \bar{f} \|_p = \left(\int dx |f(x) - \bar{f}(x)|^p\right)^{1/p}$ the term "sup" denotes supreme. These norms are denoted as C($\Re^k$); $L^1, L^2, \ldots\ldots and\ L^p$ norms, respectively. In general $L^p$ denotes the class of functions *f* such that $\|f\|_p < \infty$.

It can be said that a function $\bar{f}$ approximates another function *f*, well in $L^p$ if $\|\bar{f} - f\|$ is small.

A sequence $\{f_n\}_{n=1}^{\infty}$ converges to a function *f* in $L^p$ if $\|f_n - f\|_p \xrightarrow[n \to \infty]{} 0$.

It can be shown that the components of the above questions can be decoupled to the extent that they are equivalent to the case where there is only one *q*. Indeed, if any desired i-o function can be approximated in systems with one output neuron, such single-output systems can be easily concatenated into larger ones (with more outputs) which have essentially arbitrary approximated in input-output properties. In any case, the configuration that can be assumed is as follows (Fig 4):

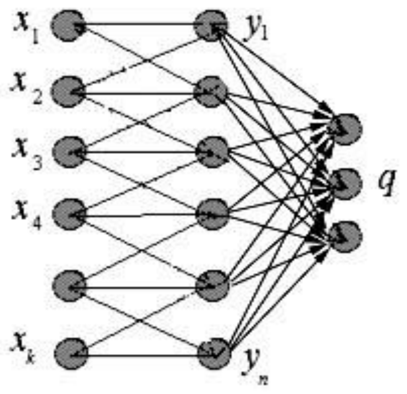


**Fig.4**

Therefore, upon the above equation we can write:

$$q_i = \sum_{j=1}^{n} w_j\, y_j = \sum_{j=1}^{n} w_j\, H(V^i . x - \theta_j) \qquad (11)$$

Consequently, for any given function *f(x)*: $\Re^k \to \Re$ , that is an approximation a partial answer which has come in the form of the solution to Hilbert's 13th problem. He proved that continuous function *f(x)*: $\Re^k \to \Re$ can be represented in the following form:

$$f(x) = \sum_{j=1}^{2k+1} \chi_j \left(\sum_{i=1}^{k} \psi_{ij}\, x_i\right) \qquad (12)$$

$\chi_j$ and $\psi_{ij}$ are continuous functions, and $\psi_{ij}$ are monotonic and independent of *f(x)*, in such a way that *f* can be represented as sum of functions that each of them depends just on a sum of single variable functions .

This artificial neural network is a form of artificial intelligence which attempt to mimic the behavior of the human brain and nervous system, (Lendaris, G., J. Neidhoefer, 2004[9]; Fausett 1994[10]). A typical structure of the feed forward neural networks consist of a number of processing elements, or nodes, that are usually arranged in layers: an input layer, an output layer and one or more hidden layers (Figure 5).

## 4. Conjugate Gradient

The general fundamental idea of conjugate gradient methods is to successively minimize the parameters $\vec{w}$ of a differentiable function *f(x):* $\Re^k \to \Re$ along a set of *k* non-interfering search directions.

If two non-interfering *conjugate* directions $\vec{v}_i$ and $\vec{v}_j$ are considered and defined in such a way that:

$$i \neq j \Rightarrow \vec{v}_i^{\,T} H \vec{v}_j = 0 \qquad (13)$$

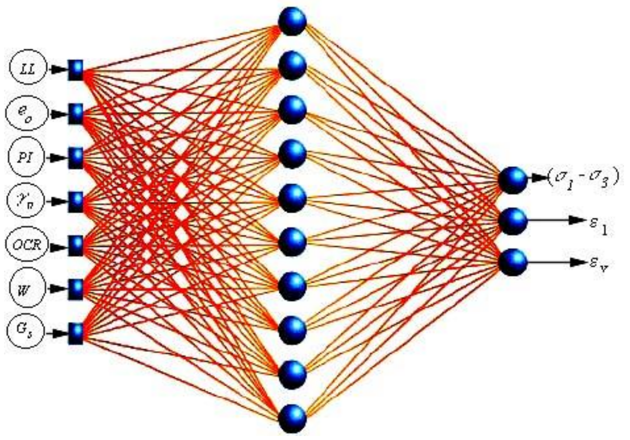


**Fig.5:** input and output layers in artificial neural networks

where $H = \nabla^2 f$ is the Hessian of the system. A set of mutually conjugate directions is produced by the following iteration as:

$$\vec{v}_t = \alpha \vec{v}_{t-1} - \vec{g}_t \;\; ; \;\; \vec{g}_t = \nabla f(\vec{w}_t) \qquad (14)$$

where $\vec{g}_t$ denotes the gradient at the $t^{th}$ iteration, and $\alpha = \left\|\vec{g}_t\right\|^2 \Big/ \left\|\vec{g}_{t-1}\right\|^2$ . A *k*-dimensional quadratic function is minimized exactly after *k* iterations of ($\vec{v}_t = \alpha \vec{v}_{t-1} - \vec{g}_t$), starting with $\vec{v}_0 = -\vec{g}_0$. This process depends upon the function being minimized exactly along each given search direction. In such a quadratic function, that is achieved by adjusting the parameters $\vec{w}$ in following form:

$$\vec{w}_{t+1} = \vec{w}_t - \beta \vec{v}_t \;\; ; \;\; \beta = \vec{g}_t^{\,T} \vec{v}_t \Big/ \vec{v}_t^{\,T} H \vec{v}_t \qquad (15)$$

For conjugate gradient on nonlinear optimization problems, an explicit linear minimization is usually employed to set the step size *β*. However, it has been shown that a local quadratic approximation (*i.e.*, the *β* given in equation 15 can be very effective here, provided that suitable trust-region modifications are made. However, a nonlinear problem will not be minimized exactly after *k* iterations, but the method can be restarted at that point by resetting $\vec{v}_t$ to the current gradient. In general, the calculation of *β* in equation 15 does *not* require explicit storage of the Hessian, which would be *O*(*k*$^2$). There are several ways to calculate the product of the Hessian with an arbitrary vector at a cost comparable to that of obtaining the gradient, which typically is *O*(*k*) (Pearlmutter, 1994). The same goes for other measures of curvature, such as the

Gauss-Newton approximation of the Hessian, and the Fisher information matrix (Schraudolph, 2002)[6]. Unfortunately conjugate gradient techniques are not designed to cope with numerical oscillation and noisy gradient; hence it tends to diverge in stochastic settings. Occasional reports to the contrary invariably concern regimes that we would term near-deterministic (i.e., large batch sizes of data). For relatively small archive data systems, the extended Kalman filter is a robust alternative, albeit at a cost of $O(k^2)$ error per iteration. Large archive data systems are therefore still often optimized using simple (first-order) gradient descent.

The convergence of simple stochastic gradient descent can be improved by adjusting a local step size for each system parameter. The most advanced of these algorithms, *stochastic meta-descent* (SMD), adapts local step sizes by a dual gradient descent procedure that uses fast curvature matrix-vector products, (Schraudolph, 1999, 2002[5, 6]).

This method adapts the system parameters $\vec{w}$ by an iterative stochastic approximation of second-order gradient steps. As formulated originally, (Orr, 1995[[11]), it was unfortunately stable only for linear systems. It has been recently succeeded in making it robust for nonlinear optimization problems by incorporating a trust-region modification (Graepel, et al, 2002[12]). This algorithm also uses fast curvature matrix-vector products.

## 5. Towards Stochastic Conjugate Gradient

To improve upon the above stochastic gradient methods by adopting certain ideas from conjugate gradient, the resulting of three new algorithms all use fast Hessian gradient products are presented.

One of the reasons why conjugate gradient breaks down in a stochastic setting is that the noise and numerical oscillation make it impossible to maintain the conjugate condition of search directions over multiple iterations, instead of trying to construct a set of conjugate directions. According to this case, let just move down the gradient as follows:

$$\vec{w}_{t+1} = \vec{w}_t - \beta \vec{g}_t \tag{16}$$

With a step size aimed at conjugating successive steps:

$$\vec{w}_{t+1} = \vec{w}_t - \beta \vec{v}_t \;;\; \beta = \vec{g}_t^{\,T} H \vec{g}_t \Big/ \left\| H \vec{g}_t \right\|^2 \tag{17}$$

This choice of $\beta$ achieves pair wise conjugation of gradients (*i.e.*, $\vec{g}_{t+1}^{\,T} H \vec{g}_t = 0$) in a deterministic quadratic function, as can easily be verified using the identity $\vec{g} = H\vec{w}$.

In systems where the Hessian has some sparsely, it is advantageous to be able to *decouple* the step sizes for individual subspaces. To this end we construct a diagonal conditioner from equation 17, by replacing the inner products by Shannon, T.T., R.A. Santiago, G. Lendaris, 2003 [13] (component-wise) products and division. In order to remove the poles resulting from zero components in the denominator, we then average both numerator and denominator over the trajectory, as follows:

$$\beta = diag\left(\frac{\left\langle diag(\vec{g}_t) H \vec{g}_t \right\rangle_\gamma}{\left\langle diag(H\vec{g}_t) H \vec{g}_t \right\rangle_\gamma}\right.\vec{g}_t \tag{18}$$

where $\langle . \rangle$ denotes exponential averaging, leads to form as follows:

$$\left\langle \vec{g}_t \right\rangle_\gamma = \vec{g}_t + \gamma \left\langle \vec{g}_{t-1} \right\rangle_\gamma \tag{19}$$

Compared to equation 17, a certain average over search space dimensions replaced with leaky averaging over the optimization trajectory.

Our third algorithm seeks to preserve as much of the conjugate gradient machinery as possible while stabilizing it for use in a stochastic setting.

As the third algorithm, to preserve as much of the conjugate gradient machinery as possible while stabilizing it for use in a stochastic setting.

As a stabilized conjugate gradient**,** the calculation of α in equation 13 is focused as a weak point.

When there are numerical oscillation in data and gradients are noisy that leads to a linear minimizations inaccurate, we could easily have $\left\|\vec{g}_t\right\| \rangle\rangle \left\|\vec{g}_{t-1}\right\|$|, resulting in inordinately large values of α. To overcome this problem we can add the correction term $\left|\vec{g}_t^{\,T} \vec{v}_{t-1}\right|$ to the denominator, and replacing equation 13, with the following:

$$\alpha = \frac{\left\|\vec{g}_t\right\|^2}{\left\|\vec{g}_{t-1}\right\|^2 + \left|\vec{g}_t^{\,T} \vec{v}_{t-1}\right|} \tag{20}$$

Note that an accurate linear minimization along $\vec{v}_{t-1}$ ensures that $\vec{g}_t^{\,T} \vec{v}_{t-1} = 0$, therefore, the above equation reduces the solution to standard conjugate gradient. The correction term comes into play only to the extent that the linear minimization fails and the gradient grows in magnitude, reducing the impact on α of this pathological situation.

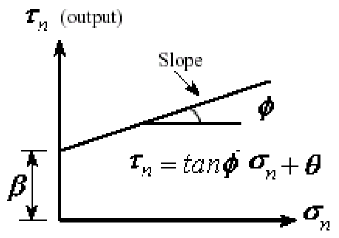
## 6. Neural Network for Mechanical Behavior of Clay

The propagation of information in artificial neural network method starts at the input layer where the input data are presented. The network adjusts its weights on the presentation of a training data set and uses a learning rule to find a set of weights that will produce the input/output mapping that has the smallest possible error. This process is called "learning" or "training".

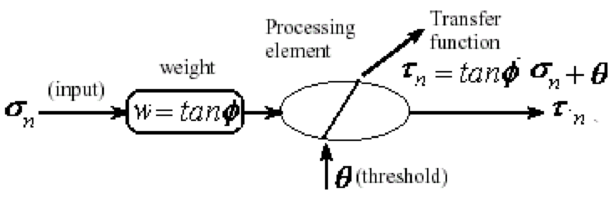
Once the training phase of the model has been successfully accomplished, the performance of the trained model has to be validated using an independent testing set. Some ordinary and specific details of this artificial neural network modeling process and development and the employed algorithms are beyond the scope of this paper and are given elsewhere (e.g. Moslehi et al. 1992[14]; Flood and Kartam 1994[15]; Maier and Dandy 2000[16]).

The feed forward artificial neural network method learns from data examples presented to them and use these data to adjust their weights in an attempt to capture the relationship between the model input variables and the corresponding outputs. Consequently, artificial neural network method does not need any prior knowledge about the nature of the relationship between the input/output variables, which is one of the benefits that artificial neural network method has compared with most empirical and statistical methods.

The artificial neural network modeling philosophy is similar to a number of conventional statistical models in the sense that both are attempting to capture the relationship between a historical set of model inputs and corresponding outputs. For example, suppose a set of *x*-values and corresponding *y*-values in 2 dimensional space, where $y = f(x)$. The objective is to find the unknown function *f*, which relates the input variable *x* to the output variable *y*. In a linear regression model, the function *f* can be obtained by changing the slope tan$\varphi$ and intercept $\beta$ of the straight line in Figure 6-a, so that the error between the actual outputs and outputs of the straight line is minimized. The same principle is used in artificial neural network models. Artificial neural network method can form the simple linear regression model by having one input, one output, no hidden layer nodes and a linear transfer function (Figure 6-b). The connection weight *w* in the artificial neural network model is equivalent to the slope tan$\varphi$ and the threshold è is equivalent to the intercept $\beta$, in the linear/nonlinear/quasi-linear regression model. Artificial neural network method adjust their weights by repeatedly presenting examples of the model inputs and outputs in order to minimize an error function between the historical outputs and the outputs predicted by the artificial neural network model.



(a)



(b)

**Fig.6:** linear regression models

However, if the relationship between $\sigma_n$ and $\tau_n$ is non-linear, regression analysis can only be successfully applied if prior knowledge of the nature of the non-linearity exists. On the contrary, this prior knowledge of the nature of the non-linearity is not required for artificial neural network models. In the artificial neural network model, the degree of non-linearity can be also changed easily by changing the transfer function and the number of hidden layer nodes. In the real world, it is likely to encounter problems that are complex and highly non-linear. In such situations, traditional regression analysis is not adequate (Gardner, 1998)[17]. In contrast, artificial neural network method can be used to deal with this complexity by changing the transfer function or network structure, and the type of non-linearity can be changed by varying the number of hidden layers and the number of nodes in each layer. In addition, artificial neural network models can be upgraded from unit-variation to multi-variation by increasing the number of input nodes.

## 7. Soil Properties and Behavior

Soil properties and behavior is an area that has attracted many researchers to modeling using artificial neural network method. Developing engineering correlations between various soil parameters is an issue discussed by Goh (1995a; 1995c)[18, 19]. Goh used neural networks to model the correlation between the relative density and the cone resistance from cone penetration test (CPT), for both normally consolidated and over-consolidated sands. Laboratory data, based on calibration chamber tests, were used to successfully train and test the neural network model. The neural network model used the relative density and the mean effective stress of soils as inputs and the CPT cone resistance as a single output. The artificial neural network model was found to give high coefficients of correlation of 0.97 and 0.91 for the training and testing data, respectively, which indicated that the neural network was successful in modeling the non-linear relationship between the CPT cone resistance and the other parameters. Many other studies have successfully used artificial neural network method for modeling soil properties and behavior, which, for brevity, are acknowledged for reference purposes in the following paragraphs.

Ellis et al. (1995)[20] developed an artificial neural network model for sands based on grain size distribution and stress history. Sidarta and Ghaboussi (1998)[21] employed an artificial neural network model within a finite element analysis to extract the geo-material constitutive behavior from non-uniform material tests. Al-Rabadi, A.N., G. Lendaris, (2003)[22] used neural networks for representing the behavior of sand and clay soils. Ghaboussi and Sidarta (1998)[23] used neural networks to model both the drained and undrained behavior of sandy soil subjected to triaxial compression-type testing. Penumadu and Zhao (1999)[24] also used artificial neural network method to model the stress-strain and volume change behavior of sand and gravel under drained triaxial compression test conditions. Zhu et al. (1998a; 1998b)[25, 26] used neural networks for modeling the shearing behavior of a fine-grained residual soil, dune sand and Hawaiian volcanic soil. Greenwood, G. W., (2005)[27] used a neural network model to generate a quantitative soil classification from three main factors (plastic index, liquid limit and clay content). Najjar et al. (1996a)[28] showed that neural network-based models can be used to accurately assess soil swelling, and that neural network models can provide significant improvements in prediction accuracy over statistical models. Greenwood, G. W., (2005)[29] showed that neural networks are able to effectively characterize and estimate the shear modulus of granular materials. Agrawal et al. (1994)[30]; Gribb and Gribb (1994)[31] and Najjar and Basheer (1996b)[32] all used neural network approaches for estimating the permeability of clay liners. Basheer and Najjar (1995)[33] presented neural network approaches for soil compaction.

Other applications include modeling the mechanical behavior of medium-to-fine sand (Ellis et al. 1992)[34], modeling rate-dependent behavior of clay soils (Penumadu et al. (1994)[35], simulating the uniaxial stress-strain constitutive behavior of fine-grained soils under both monotonic and cyclic loading (Basheer 1998)[36], characterizing the undrained stress-strain response of Nevada sand subjected to both triaxial compression and extension stress paths (Najjar and Ali 1999[37]), predicting the axial and volumetric stress-strain behavior of sand during loading, unloading and reloading (Zhu and Zaman 1997)[38], predicting the anisotropic stiffness of granular materials from standard repeated load triaxial tests (Tutumluer and Seyhan 1998)[39].

## 8. Input Data Structure

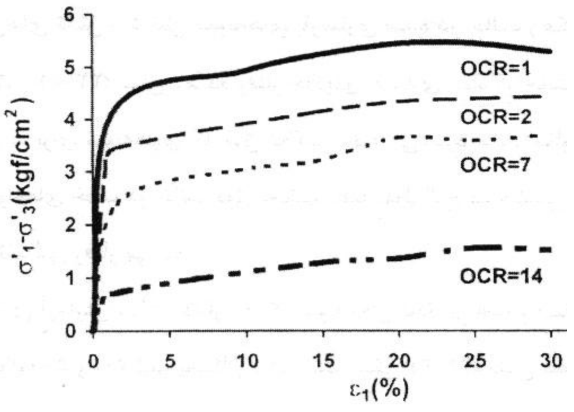
The learning rate parameter first is defined as $\alpha$. Any higher value of this parameter increases the learning speed of the model. However, this condition may lead the solution to lack of convergence. A recommended value of $\alpha$ is 0.001.

The input data structure of each learning sample data includes eight sample parameters as $e_o$ (initial void ratio), LL (liquid limit), PI (plasticity index), $G_s$ (solid grain density), $\omega$ (moisture content), $\gamma_n$ (natural soil density), OCR (over consolidation ratio), and $\sigma_{3c}$ (initial mean stress). The first line of each data layer includes these eight parameters. The second line includes the number of measured stress-strain sets during the performed test. The final value of strain in the test is an optional value.

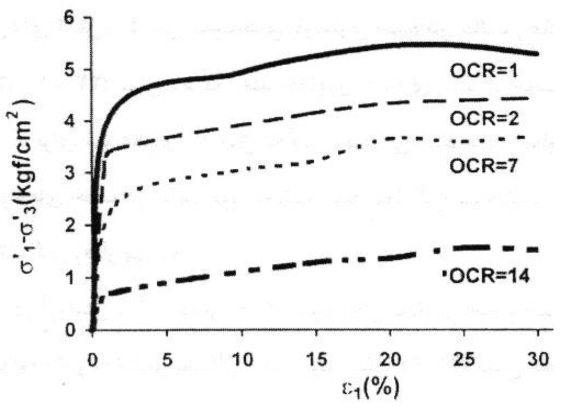
The weighted coefficients in initial weighted coefficients matrix and initial bias vector are quite arbitrary and even may be taken as constant values. The learning algorithm is planed based on the following procedure:

- Reading data
- Internal/external proportionality of strain measured values
- Homogenization of data
- Introducing initial weighted coefficient matrix and bias vector as preferred
- BBP calculations based of proceeding and returning back
- Error control based on any applicable method such as root-square method
- Continuing in the case of error control or returning to BBP for back ward algorithm with new adjusted parameter values
- Transferring data to initial values
- Presentation of output data

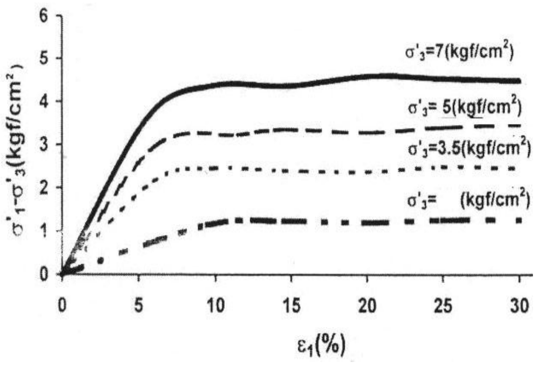
The learning information include standard triaxial test results of CD for q (stress deviator) versus $\varepsilon_1$ (axial strain) for different OCR values as shown in Figure 7-a,b and also CD tests on disturbed samples under different initial consolidation mean stress as shown in Figure 7-c. This learning information include some CU tests results as q (stress deviator) versus $\varepsilon_1$ (axial strain) on disturbed samples as shown in Figures 8-a,b,c similar to Figures 7-a,b,c.



(a)

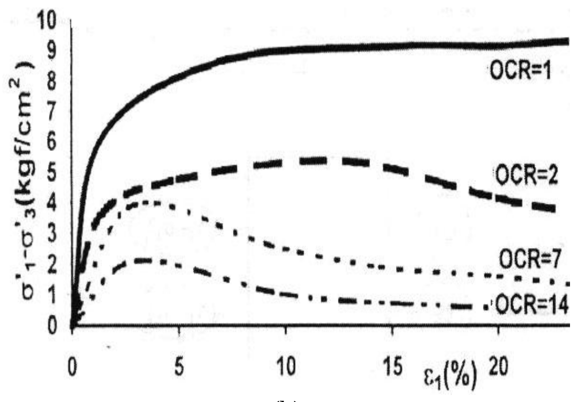


(b)

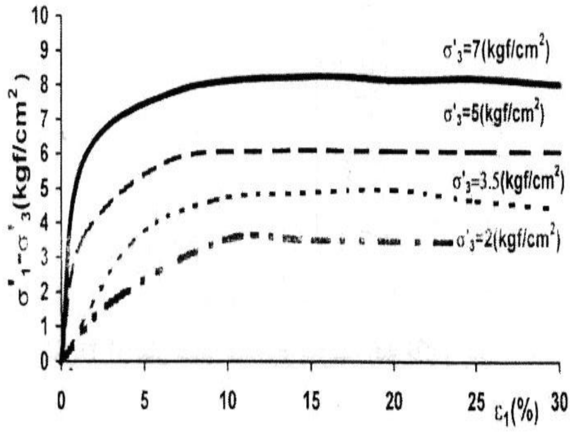


(c)

**Fig.7:** learning information of CD tests



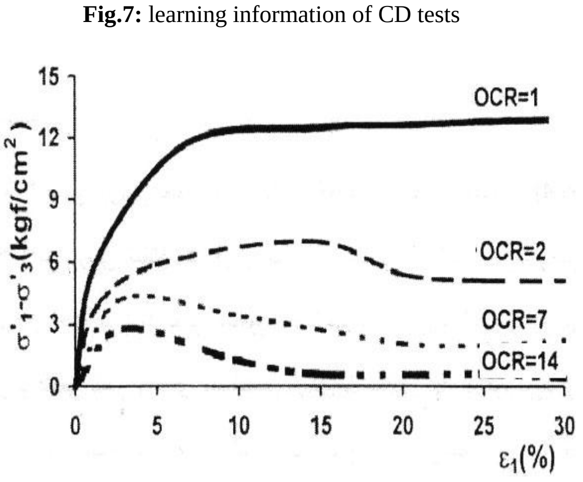
(b)



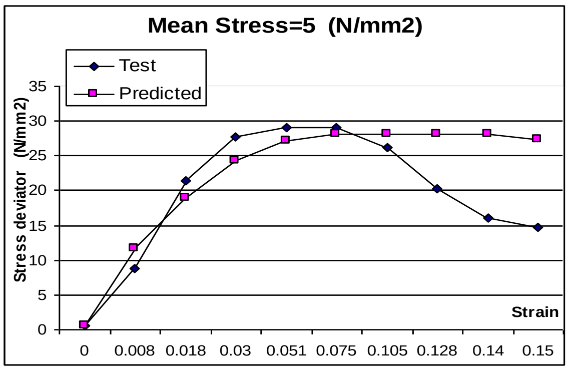
(c)

**Fig.8:** learning information of CU tests

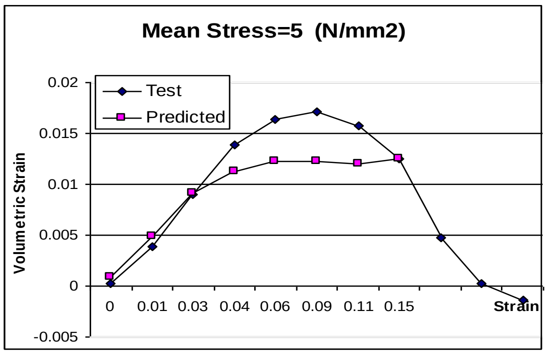
The verified results obtained based on the employed learning information include also some standard triaxial CD test results as q (stress deviator) versus $\varepsilon_1$ (axial strain) and $\varepsilon_v$ (volumetric strain) versus $\varepsilon_1$ (axial strain) for different OCR values as shown in Figure 9. This presented information include also some CU tests results as q (stress deviator) versus $\varepsilon_1$ (axial strain) and P.W.P (pore water pressure) versus $\varepsilon_1$ (axial strain) on disturbed samples upon different mean stresses as shown in Figures 10.



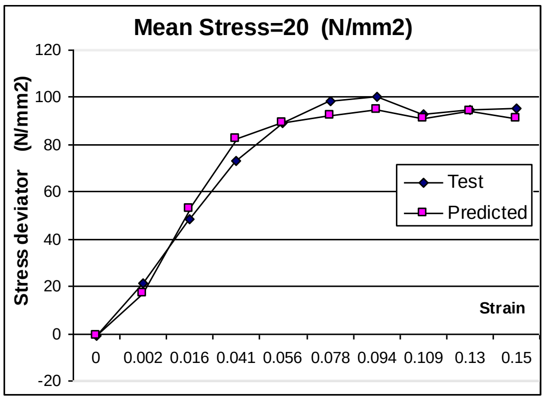
(a)



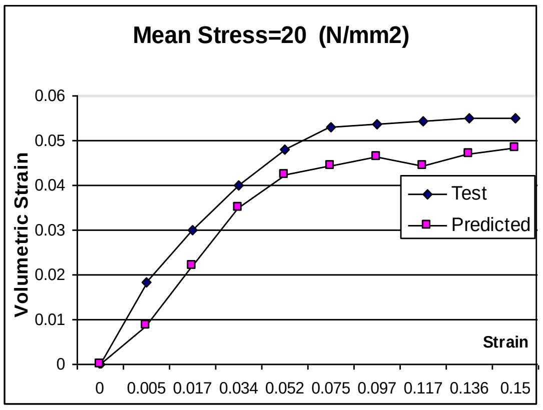
(a)



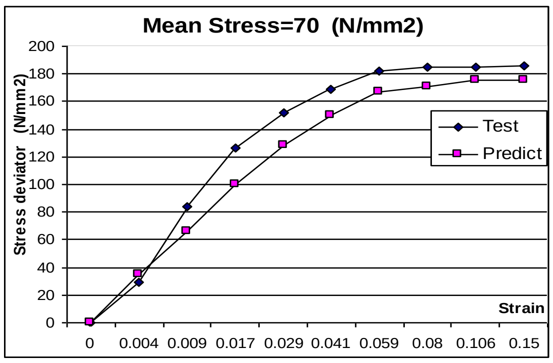
(b)



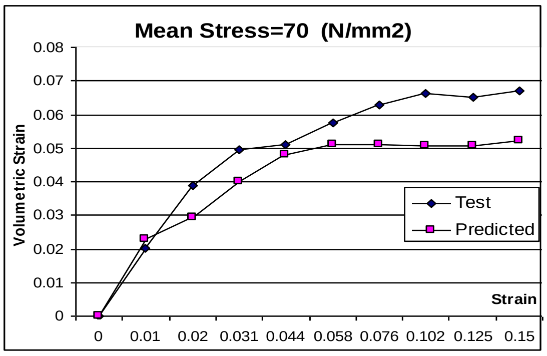
(c)



(d)

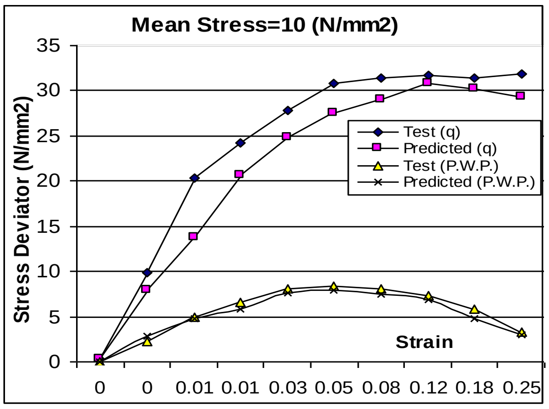


(e)

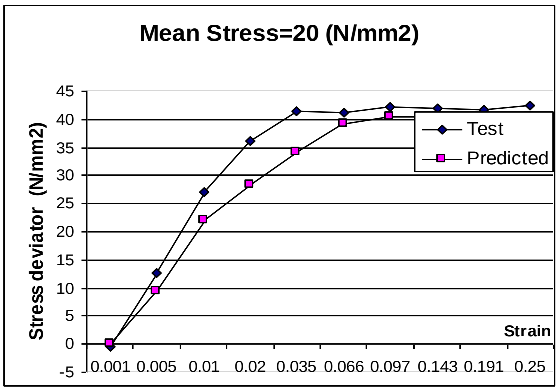


(f)

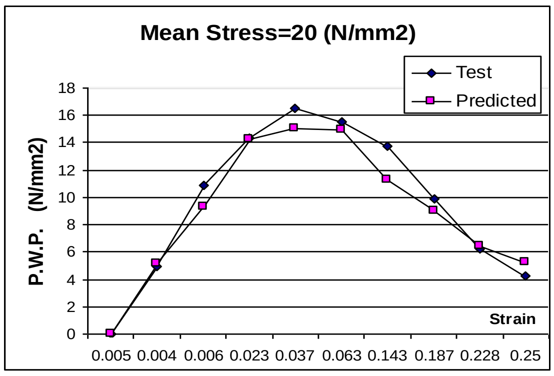
**Fig.9:** CD test results



(a)



(b)



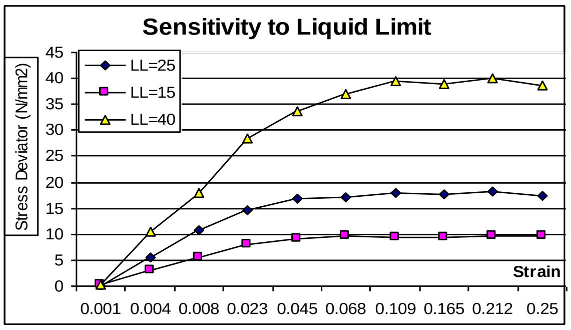
(c)

**Fig.10:** CU test results

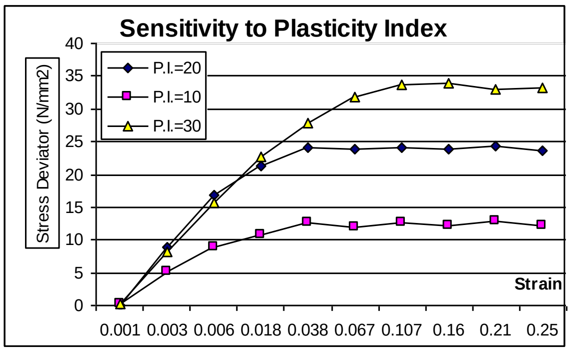
## 9. Sensitivity Investigation

In artificial neural network method the output parameter sensitivity to any of input parameters can be investigated. The results of this investigation may present the mathematical relation between every of input and output parameters while the other parameters are kept constant or non-effective. To show this ability the effects of liquid limit and plasticity index on shear strength are investigated. Figure 11-a, b show the variation of shear strength due to changes of liquid limit and plasticity index respectively. Accordingly, equation 21 presents strength change versus plasticity index.

$$C_u / P_o = 0.11 + 0.0001PI \qquad (21)$$



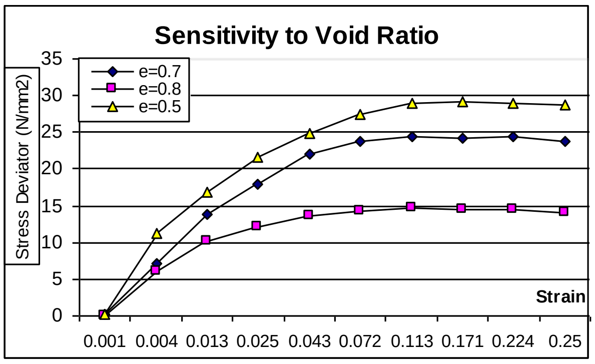
(a)



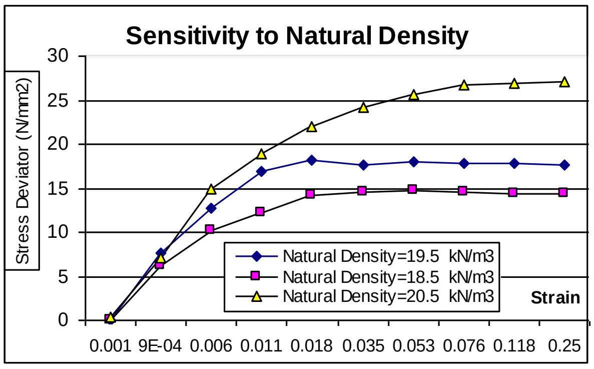
(b)

**Fig.11:** variation of shear strength due to changes of liquid limit and plasticity index

In similar investigation by authors, it is found that there is no serious change of shear strength in stress-strain curve due to grain density and moisture content in this case. This condition may be obtained because there has not been any information of grain density and moisture content effects in teaching process. Figures 12-a, b show the effects of natural density and initial void ratio on stress-strain curve respectively. Accordingly, increment of natural density and reduction in initial void ratio increase shear strength.



(a)



(b)

**Fig.12-a, b:** the effects of natural density and initial void ratio

Also, there is not any sensitivity on OCR ratio, because there has not been any learning of this parameter as input information. However, the effects of lateral stress ($\sigma_3$) is shown in Figure 12-c., naturally, the increment of lateral pressure caused increases in shear strength.

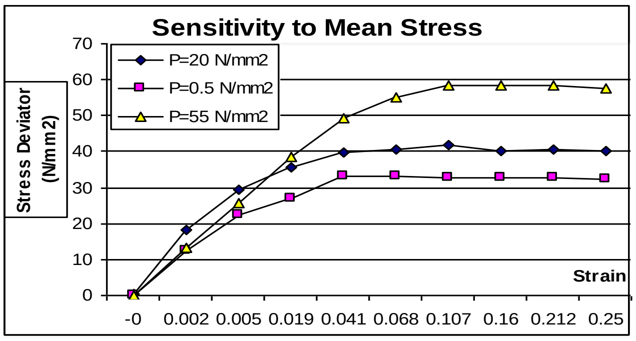


**Fig.12-c:** the effects of lateral pressure on shear strength.

## 10. Conclusion

New form of neural network technique is developed and tested them in predicting cohesive soil behavior upon quadratic function. The results are quite promising in that one of our techniques outperforms simple stochastic gradient by an order of magnitude in all settings.

We also hope to further improve upon these techniques through a systematic re-derivation of the conjugate gradient method in the linear stochastic setting in future.

Upon geo-mechanical investigation, eight mechanical properties of soil as liquid limit, plasticity index, void ratio, natural density, moisture content, OCR, and lateral stress are selected as input parameters to predict stress-strain curve for south Tehran clay. The input test data were from triaxial standard CU and CD tests.

It is evident from the results that a certain artificial neural network method has been applied successfully to predict clay behavior. However, accuracy of the predicted results is highly related to learning process as well as the employed algorithm. Any more conformity of active behavioral aspects in teaching process to output aspects, a better accuracy is obtained. Despite of interrelation of learning input parameter effects, any new aspect effects can be easily added in learning process. Furthermore, the capability of artificial neural network in learning is quite high. In this artificial neural network method the output parameter sensitivity to any of input parameters can be investigated. This capability is a rational way to investigate the relation between any of two dependent/independent parameters in certain loading process.

There are also several areas in which the feasibility of artificial neural network method has yet to be tested. In many situations in geotechnical engineering, it is possible to encounter some types of problems that are very complex and not well understood that can be solved by neural network method. For most mathematical models that attempt to solve such problems, the lack of physical understanding is usually supplemented by either simplifying the problem or incorporating several assumptions into the models. Mathematical models also rely on assuming the structure of the model in advance, which may be sub-optimal. Consequently, many mathematical models fail to simulate the complex behavior of most geotechnical engineering problems. In contrast, artificial neural network method is based on the data alone in which the model can be trained on input-output data pairs to determine the structure and parameters of the model. In this case, there is no need to neither simplify the problem nor incorporate any assumptions. Moreover, artificial neural network method can always be updated to obtain better results by presenting new training examples as new data become available.

Despite their good performance in many situations, artificial neural network method suffer from a number of shortcomings, notably, the lack of theory to help with their development, the fact that success in finding a good solution is not always guaranteed and their limited ability to explain the way they use the available information to arrive a solution. Consequently, there is a need to develop some guidelines, which can help in the design process of artificial neural network method. There is also a need for more research to give a comprehensive explanation of how artificial neural network method arrives at a prediction.

## References

**[1]** Jaksa, M. B. (1995). “The influence of spatial variability on the geotechncial design properties of a stiff, overconsolidated clay,” PhD thesis, The University of Adelaide, Adelaide.

**[2]** Hubick, K. T. (1992). *Artificial neural networks in Australia*, Department of Industry, Technology and Commerce, Commonwealth of Australia, Canberra.

**[3]** Bhagat, P.M. (2005) *Pattern Recognition in Industry*, Elsevier. ISBN 0-08-044538-1

**[4]** Marquardt.D., (1963), An algorithm for least-squares estimation of non-linear parameters. *Journal of the Society of Industrial and Applied Mathematics*, 11(2):431–441.

**[5]** Schraudolph, N.N., (1999), Local gain adaptation in stochastic gradient descent. In *Proceedings of the 9th International Conference on Artificial Neural Networks*, pages 569–574, Edinburgh, Scotland, 1999. IEE, London.

**[6]** Schraudolph, N.N., (2002), Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738.

**[7]** Santiago, R.A., G. Lendaris, (2005), “Reinforcement Learning and the Frame Problem,” *Proc. IJCNN.*

**[8]** Funahashi. K. (1989), On the approximate realization of continuous mappings by neuralnetworks, Neural Networks 2, 183-192, 1989 .

**[9]** Lendaris, G., J. Neidhoefer, (2004), “Guidance in the Use of Adaptive Critics for Control,” Ch. 4 in *Handbook of Learning and Approximate Dynamic Programming*, J. Si, A.G. Barto, W.B. Powell, D. Wunsch, Eds., 97-124.

**[10]** Fausett, L. V. (1994). *Fundamentals neural networks: Architecture, algorithms, and applications*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

**[11]** Orr, G. B., (1995), *Dynamics and Algorithms for Stochastic Learning*. PhD thesis, Department of Computer Science and Engineering, Oregon Graduate Institute, Beaverton, OR 97006, 1995. ftp://neural.cse.ogi.edu/pub/neural/papers/orrPhDch1-5. ps.Z, orrPhDch6-9.ps.Z.

**[12]** Graepel, T. and Schraudolph, N. N.(2002), Stable adaptive momentum for rapid online learning in nonlinear systems. In Dorronsoro [11]. http://www.inf.ethz.ch/~schraudo/pubs/sam.ps.gz.

**[13]** Shannon, T.T., R.A. Santiago, G. Lendaris, (2003), Accelerated Critic Learning in Approximate Dynamic Programming via Value Templates and Perceptual Learning, *Proc. IJCNN.*

**[14]** Moselhi, O., Hegazy, T., and Fazio, P. (1992). “Potential applications of neural networks in construction.” *Can. J. Civil Engrg*, 19, 521-529.

**[15]** Flood, I., and Kartam, N. (1994). “Neural networks in civil engineering I: Principles and understanding.” *J. Computing in Civil Engrg*, ASCE, 8(2), 131-148.

**[16]** Maier, H. R., and Dandy, G. C. (2000). “Neural networks for the prediction and forecasting of water resources variables: A review of modelling issues and applications.” *Environmental* Modelling & Software, 15(2000), 101-124.

**[17]** Gardner, M. W., and Dorling, S. R. (1998). "Artificial neural networks (The multilayer perceptron) - A review of applications in the atmospheric sciences." *Atmospheric Environment*, 32(14/15), 2627-2636.

**[18]** Goh, A. T. C. (1994a). "Nonlinear modelling in geotechnical engineering using neural networks." *Australian Civil Engineering Transactions*, CE36(4), 293-297.

**[14]** Goh, A. T. C. (1995b). "Empirical design in geotechnics using neural networks." *Geotechnique*, 45(4), 709-714.

**[15]** Goh, A. T. C. (1995c). "Modeling soil correlations using neural networks." *J. Computing in Civil Engrg.*, ASCE, 9(4), 275-278.

**[16]** Goh, A. T. C. (1996a). "Neural-network modeling of CPT seismic liquefaction data." *J. Geotech. Engrg.*, ASCE, 122(1), 70-73.

**[18]** Hestenes, M.R. and Stiefel, E., (1952), Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

**[19]** Goh, A. T. C. (1995a). "Back-propagation neural networks for modeling complex systems." *Artificial Intelligence in Engineering*, 9, 143-151.

**[20]** Ellis, G. W., Yao, C., Zhao, R., and Penumadu, D. (1995). "Stress-strain modelling of sands using artificial neural networks." *J. Geotech. Engrg.*, ASCE, 121(5), 429-435.

**[21]** Sidarta, D. E., and Ghaboussi, J. (1998). "Constitutive modeling of geomaterials from non-uniform material tests." *J. Computers & Geomechanics*, 22(10), 53-71.

**[22]** Al-Rabadi, A.N., G. Lendaris, (3003), Artificial Neural Network Implementation Using Many-Valued Quantum Computing, *Proceedings of IJCNN.*

**[23]** Ghaboussi, J., and Sidarta, D. E. (1998). "New nested adaptive neural networks (NANN) for constitutive modeling." *J. Computers and Geotechnics*, 22(1), 29-52.

**[24]** Penumadu, D., and Zhao, R. (1999). "Triaxial compression behavior of sand and gravel using artificial neural networks (ANN)." *J. Computers and Geotechnics*, 24, 207-230.

**[25]** Zhu, J. H., Zaman, M. M., and Anderson, S. A. (1998a). "Modeling of soil behavior with a recurrent neural network." *Canadian Geotech. J.*, 35(5), 858-872.

**[26]** Zhu, J. H., Zaman, M. M., and Anderson, S. A. (1998b). "Modelling of shearing behavior of a residual soil with recurrent neural network." *Int. J. Numerical and Analytical Methods in Geomechanics*, 22(8), 671-687.

**[27]** Greenwood, G. W., (2005), On the practicality of using intrinsic reconfiguration for fault recovery, *IEEE Transactions on Evolutionary Computation* 9(4), 398-405.

**[28]** Najjar, Y. M., and Basheer, I. A. (1996a). "Neural network approach for site characterization and uncertainty prediction." *Geotechnical Special Publication*, ASCE, 58(1), 134-148.

**[29]** Greenwood, G. W., (2005), ``On the usefulness of accessibility graphs with combinatorial optimization problems", *Journal of Interdisciplinary Mathematics* 8(2), 277-286.

**[30]** Agrawal, G., Weeraratne, S., and Khilnani, K. (1994). "Estimating clay liner and cover permeability using computational neural networks." *Proc., First Congress on Computing in Civil Engrg.*, Washington, June 20-22.

**[31]** Najjar, Y. M., Ali, H. E., and Basheer, I. A. (1999). "On the use of neurons for simulating the stress-strain behavior of soils." *Proc., 7th Int. Symposium on Numerical Models in Geomechanics*, G. N. Pande, ed., Graz, Austria, NUMOG VII, September 1-3, 657-662.

**[32]** Najjar, Y. M., and Basheer, I. A. (1996b). "Utilizing computational neural networks for evaluating the permeability of compacted clay liners." *Geotechnical and Geological Engineering*, 14, 193-221.

**[33]** Basheer, I. A., and Najjar, Y. M. (1995). "A neural-network for soil compaction." *Proc., 5th Int. Symp. Numerical Models in Geomechanics*, G. N. Pande and S. Pietruszczak, eds., Roterdam: Balkema, 435-440.

**[34]** Ellis, G. W., Yao, C., and Zhao, R. (1992). "Neural network modeling of the mechanical behavior of sand." *Proc., Engineering Mechanics*, ASCE, 421-424.

**[35]** Pearlmutter, B. A., (1994), Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160.

**[35]** Penumadu, D., Jin-Nan, L., Chameau, J.-L., and Arumugam, S. (1994). "Rate dependent behavior of clays using neural networks." *Proc., 13th Conf. Int. soc. Soil Mech. & Found. Engrg.*, New Delhi, 4, 1445-1448.

**[36]** Basheer, I. A. (1998). "Neuromechanistic-based modeling and simulation of constitutive behavior of fine-grained soils." Ph.D. dissertation, KansasStateUniversity, Manhattan, KS.

**[37]** Najjar, Y. M., and Ali, H. E. (1999). "Simulating the stress-strain behavior of Nevada sand by ANN." *Proc., 5$^{th}$ U.S.National Congress on Computational Mechanics (USACM)*, Boulder, Colorado, August 4-6.

**[38]** Zhu, J. H., and Zamman, M. M. (1997). "Neural network modeling for a cohesionless soil." *76th Meeting of the Transportation Research Board*, January, Washington, D.C.,

**[39]** Tutumluer, E., and Seyhan, U. (1998). "Neural network modeling of anisotropic aggregate behavior from repeated load triaxial tests. " *Transportation Research Record 1615*, National Research Council, Washington, D.C.