Working Paper

Artificial Neural Network Representations for Hierarchical Preference Structures

Antonie Stam
Minghe Sun
Marc Haines

WP-95-33
April 1995

IIASA International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 807 □ Fax: +43 2236 71313 □ E-Mail: info@iiasa.ac.at

Artificial Neural Network Representations for Hierarchical Preference Structures

Antonie Stam
Minghe Sun
Marc Haines

WP-95-33
April 1995

International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 807 □ Fax: +43 2236 71313 □ E-Mail: info@iiasa.ac.at

ARTIFICIAL NEURAL NETWORK REPRESENTATIONS FOR HIERARCHICAL PREFERENCE STRUCTURES

Antonie Stam[1, 2, 5]

Minghe Sun[3]

Marc Haines[4]

March 6, 1995

1: Methodology of Decision Analysis Project
International Institute for Applied Systems Analysis
A-2361 Laxenburg, Austria

2: Department of Management
Terry College of Business
The University of Georgia
Athens, GA 30602, U.S.A.

3: Division of Management and Marketing
College of Business
The University of Texas at San Antonio
San Antonio, TX 78249, U.S.A.

4: Computer Science Department
University of Koblenz-Landau
Rheinau 1
56075 Koblenz, Germany

5: Corresponding author.

FOREWORD

In real-life decision problems, the preference statements elicited from a decision maker are bound to be somewhat imprecise. Most traditional decision analytic tools do not take this impreciseness into account explicitly. It is well-known that neural networks have been used successfully for solving pattern recognition problems in the presence of incomplete, fuzzy or imprecise information. Recognizing that preference structures can be viewed as patterns, this paper introduces two different neural network representations for predicting a decision maker's ratio-scale preference ratings from matrices of pairwise preference jugdgments, and compares the accuracy of these predicted ratings with that of the power method used in AHP, a widely used discrete alternative multicriteria method for rating alternatives in the presence of qualitative criteria. A limited simulation experiment shows that, in the presence of imprecise preference information, feed-forward neural network models can indeed provide more accurate ratings than the power method. As the neural network approach to predicting ratings can easily be embedded in any software implementation of the AHP, the findings of this paper appear to represent a promising contribution to the practice of decision making.

ARTIFICIAL NEURAL NETWORK REPRESENTATIONS FOR HIERARCHICAL PREFERENCE STRUCTURES

ABSTRACT

In this paper, we introduce two artificial neural network formulations that can be used to predict the preference ratings from the pairwise comparison matrices of the Analytic Hierarchy Process (AHP). First, we introduce a modified Hopfield network that can be used to exactly determine the vector of preference ratings associated with a positive reciprocal comparison matrix. The dynamics of this network are mathematically equivalent to the power method, a widely used numerical method for computing the principal eigenvectors of square matrices. However, we show that the Hopfield network representation is incapable of generalizing the preference patterns, and consequently is not suitable for approximating the preference ratings if the preference information is imprecise. Then we present a feed-forward neural network formulation that does have the ability to accurately approximate the preference ratings. A simulation experiment is used to verify the robustness of the feed-forward neural network formulation with respect to imprecise pairwise judgments. From the results of this experiment, we conclude that the feed-forward neural network formulation appears to be a powerful tool for analyzing discrete alternative multicriteria decision problems with imprecise or fuzzy ratio-scale preference judgments.

Keywords: Decision Analysis, Multicriteria Decision Making, Artificial Neural Networks, Analytic Hierarchy Process.

ARTIFICIAL NEURAL NETWORK REPRESENTATIONS FOR HIERARCHICAL PREFERENCE STRUCTURES

1. INTRODUCTION

In this paper, we introduce two artificial neural network (*ANN*) representations of hierarchical preference structures, in particular as these relate to the Analytic Hierarchy Process (AHP) (Saaty 1988). An *ANN* is a set of processing units or computational elements, called nodes, connected with links, called arcs, with connectivity weights representing the strength of connections. We first show that the dynamics of a modified Hopfield neural network formulation are identical to the power method of computing the principal eigenvalues and principal eigenvectors of square matrices, and hence to the calculation of the vector of preference ratings of the AHP. Since this *ANN* representation cannot generalize preference information, it cannot produce accurate results when used for approximating preference structures where the decision maker's (DM's) preference judgments are imprecise. Next, we introduce a feed-forward *ANN* formulation that is capable, in the presence of imprecise judgments, of approximating the mapping from any positive reciprocal pairwise comparison matrix to its preference rating vector more accurately than the principal eigenvector. Through a simulation experiment, we verify that the trained feed-forward *ANN*s yield preference ratings that are robust with respect to perturbations in the pairwise preference judgments. Thus, the feed-forward *ANN* formulation is a powerful tool for analyzing discrete alternative multicriteria problems, in particular if the pairwise preference judgments are imprecise.

This paper is organized as follows. Section 2 gives a brief review of the AHP and a summary of the power method. In Section 3 we show how a modified Hopfield artificial neural network can be used to exactly determine the principal eigenvectors of the pairwise comparison matrices, in the same way as the power method does. In Section 4, we introduce a feed-forward *ANN* formulation that can be used to approximate AHP preference ratings, describe how the *ANN*s were trained, and provide information regarding *ANN* architecture. Section 5 details the results of a simulation study that illustrates the generalizing ability of the feed-forward *ANN* formulation, rendering it very suitable for approximating the AHP preference ratings if the pairwise preference information provided by the DM is imprecise. The paper concludes with final remarks in Section 6.

2. THE ANALYTIC HIERARCHY PROCESS

2.1. Background and Concepts

The AHP (Saaty 1988) is a widely used method for analyzing complex discrete alternative decision problems with multiple qualitative criteria. In the AHP, the decision problem is decomposed into a tree-like hierarchical structure, with the overall goal on the top and the discrete alternatives at the bottom. The intermediate levels of the hierarchy represent lower level criteria that contribute to the overall goal.

A rationale of the AHP is that decomposing a complex decision problem into more easily managed smaller-scale sub-problems, and sequentially evaluating the trade-offs between alternatives for these sub-problems, facilitates a meaningful analysis of the overall problem. At each level of the hierarchy, the DM evaluates the relative importance or attractiveness of each pair of elements (alternatives or criteria) with respect to elements (criteria) at the next higher level of the hierarchy. The process of eliciting preference judgments proceeds in a top-down fashion, one level at a time, from the overall goal to the lowest level of the hierarchy, and results in a set of positive reciprocal pairwise comparison matrices. The top-down preference elicitation requires the DM's direct involvement at every stage. The principal eigenvalues and the principal eigenvectors of these reciprocal comparison matrices are then computed. The AHP is based on the premise that the principal eigenvector of a matrix of pairwise comparisons is an accurate measure of the relative importance or attractiveness of the elements (alternatives or criteria) with respect to an element at the next higher level, and interprets the normalized principal eigenvector components as preference ratings of the elements (Saaty 1977, 1988). Saaty also uses the principal eigenvalue to develop a measure of consistency of the DM's pairwise comparison judgments (Saaty 1988).

After preference elicitation, a bottom-up approach is used to aggregate (combine) the eigenvectors of the pairwise comparison matrices into composite preference ratings of the alternatives, starting with an aggregation of their relative attractiveness with respect to the lowest level criteria. In contrast with the top-down process of eliciting preferences, the bottom-up preference aggregation is purely mechanical, and does not involve the DM's participation. The sequential aggregation of the normalized principal eigenvectors at all levels of the hierarchy results in global preference ratings of the alternatives with respect to the overall objective. Clearly, the major numerical effort in the AHP involves computing the principal eigenvectors and the principal eigenvalues of the pairwise comparison matrices. We refer the reader to Saaty (1988) for the computational details.

In the remainder of this paper, we will assume that the decision problem has one criterion level. This assumption merely simplifies the notation, and does not represent any limitation to the methodology outlined below. In the case of multiple criterion levels, the sub-problems are to be

aggregated sequentially across the levels of the hierarchy.

In the literature, there has been considerable criticism of this original AHP scale, in terms of its range, as well the choice of discrete points within the range, and the relevance and interpretation of the scale itself (Dyer 1990; Pöyhönen, Hämäläinen and Salo 1994; Schoner and Wedley 1989; Winkler 1990). The use of the principal eigenvector as a measure of relative preference has been questioned as well (Barzilai, Cook and Golani 1987). Even though these are interesting research topics, these issues fall outside the scope of this paper, and we will not discuss them in detail.

We will adopt the following notation. Denote the relative pairwise preference of alternative i over alternative j with respect to criterion m by a_{ijm}. In the AHP, the range of values of a_{ijm} is limited to [1/9, 9]. The reciprocity assumption of the AHP implies that $a_{ijm} = 1/a_{jim}$, for all i, j. Among others, this implies that $a_{iim} = 1$, for all i. Define the matrix of pairwise comparisons with respect to criterion m by $\mathbf{A}_m = (a_{ijm})$. To simplify the notation, we will omit the subscript m, so that $a_{ijm} = a_{ij}$, and $\mathbf{A}_m = \mathbf{A}$.

2.2. The Power Method for Computing the Principal Eigenvector and the Principal Eigenvalue

Let λ_1, ..., λ_n and $\mathbf{v}_1$, ..., $\mathbf{v}_n$ be the eigenvalues and eigenvectors, respectively, of an $n \times n$ matrix $\mathbf{A}$, with $|\lambda_1| \geq ... \geq |\lambda_n|$. An analytical solution of the eigenvalues of $\mathbf{A}$ requires the determination of the n roots of the characteristic polynomial $|\mathbf{A} - \lambda\mathbf{I}| = 0$. The eigenvector $\mathbf{v}_j$ corresponding to λ_j is then found by solving $\mathbf{A}\mathbf{v}_j = \lambda_j\mathbf{v}_j$. If n is large, the analytical method is computationally prohibitive, and numerical methods are used instead. Many applications, including the AHP (Saaty 1977, 1988), require only the principal eigenvalue λ_1 and the principal eigenvector $\mathbf{v}_1$. In such cases, the power method (Burden and Faires 1989; Cohen and *et al.* 1973) can be used to numerically compute λ_1 and $\mathbf{v}_1$.

For any eigenvalue λ_j and associated eigenvector $\mathbf{v}_j$ of $\mathbf{A}$, $\mathbf{A}\mathbf{v}_j = \lambda_j\mathbf{v}_j$ holds, so that $\mathbf{A}^k\mathbf{v}_j = \lambda_j^k\mathbf{v}_j$, for any $1 \leq j \leq n$ and for any positive integer k. Since the $\mathbf{v}_j$ are linearly independent, any arbitrary vector $\mathbf{v}^{(0)} \in \Re^n$ can be written as a linear combination of $\mathbf{v}_1$, ..., $\mathbf{v}_n$:

$$\mathbf{v}^{(0)} = \sum_{j=1}^{n} \alpha_j\mathbf{v}_j, \tag{2.1}$$

where not all scalars α_j are equal to zero. Pre-multiplying both sides of (2.1) by $\mathbf{A}^k$ yields (2.2),

$$\mathbf{A}^k\mathbf{v}^{(0)} = \sum_{j=1}^{n} \alpha_j\mathbf{A}^k\mathbf{v}_j = \sum_{j=1}^{n} \alpha_j\lambda_j^k\mathbf{v}_j$$

$$= \alpha_1\lambda_1^k\mathbf{v}_1 + \lambda_1^k\sum_{j=2}^{n} \alpha_j\left(\frac{\lambda_j}{\lambda_1}\right)^k\mathbf{v}_j. \tag{2.2}$$

If $|\lambda_1| > |\lambda_2|$, then $\lim_{k\to\infty}\left(\frac{\lambda_j}{\lambda_1}\right)^k = 0$, for $j = 2$, ..., n, so that the limit of (2.2) as k tends to ∞ can

be written as (2.3),

$$\lim_{k\to\infty} \mathbf{A}^k \mathbf{v}^{(0)} = \lim_{k\to\infty} \alpha_1 \lambda_1^k \mathbf{v}_1. \tag{2.3}$$

For meaningful results, the limit of (2.3) should be finite, but not zero. Hence, the vector $A^k\mathbf{v}^{(0)}$ in (2.2) should be normalized for proper convergence (Burden and Faires 1989). This is achieved by computing the vector $\boldsymbol{net}^{(k)}$ in (2.4) at each iteration k:

$$\boldsymbol{net}^{(k)} = \mathbf{A}\mathbf{v}^{(k-1)}, \tag{2.4}$$

and normalizing $\boldsymbol{net}^{(k)}$ to obtain a new vector $\mathbf{v}^{(k)}$ with, depending on whether the L_1- or L_∞-norm is used in the normalization, the components $v_{i,1}^{(k)}$ or $v_{i,\infty}^{(k)}$ in (2.5):

$$v_{i,1}^{(k)} = \frac{net_i^{(k)}}{\sum_{j=1}^{n} net_j^{(k)}}, \text{ or } v_{i,\infty}^{(k)} = \frac{net_i^{(k)}}{\max\limits_{1 \le j \le n} \{|net_j^{(k)}|\}}. \tag{2.5}$$

If λ_1 exists, $\mathbf{v}^{(k)}$ converges to $\mathbf{v}_1 = (v_{1i}, \ldots, v_{1n})^T$, and $\sum_{j=1}^{n} net_j^{(k)}$ or $\max\limits_{1 \le j \le n} \{|net_j^{(k)}|\}$ converges to λ_1 as k increases (Cohen and *et al.* 1973; Burden and Faires 1989).

The power method starts with an arbitrary vector $\mathbf{v}^{(0)} \in \Re^n$, with $\alpha_1 \neq 0$, and alternately applies (2.4) and (2.5), until either $\mathbf{v}^{(k)}$ stabilizes within a pre-specified tolerance level or a pre-specified number of iterations has been performed. In the latter case, $\mathbf{v}_1$ and λ_1 may not exist.

Denote the normalized principal eigenvector by $\mathbf{r}$, where $r_i = v_{1i}/\Sigma_{j=1}^{n} v_{1j}$. Saaty (1988) poses that, if the a_{ij} values indeed represent the true preference structure, the ratio r_i/r_j exactly equals the "true" relative preference of alternatives i and j, *i.e.*, $a_{ij} = r_i/r_j$, so that the vector $\mathbf{r}$ is interpreted as a measure of the true preference rating vector, and r_i represents the rating of alternative i. We will adopt this interpretation throughout this paper, and – unless stated otherwise – will refer to the normalized principal eigenvector $\mathbf{r}$ as the "true" rating vector. However, it is important to keep in mind that in practice the a_{ij} will be approximate, and within the AHP framework $\mathbf{r}$ will be a proxy for the true preference ratings. The fact that elicited preferences are usually approximate was a major motivation for using *ANN*s to represent these structures.

3. A MODIFIED HOPFIELD NETWORK FORMULATION FOR THE AHP: EXACT PREFERENCE RATINGS

3.1. The Hopfield Neural Network

We will only describe the Hopfield network to the extent needed to explore its relationship with the power method, as applied to the AHP. In a Hopfield net, all artificial neurons (also called nodes or units) are fully connected by bi-directional arcs (artificial synapses) (Hopfield 1982, 1984). Denote the connectivity weight of the arc from node j to node i by w_{ij}, and the matrix of connectivity weights by $\mathbf{W} = (w_{ij})$. Let $net_i^{(k)}$ and $v_i^{(k)}$ represent the input and output of neuron i at iteration k, respectively. The state of an n-neuron Hopfield network at iteration k is defined as the output vector $\mathbf{v^{(k)}} = (v_1^{(k)}, \ldots, v_n^{(k)})^{\mathrm{T}} \in \Re^n$. In the original Hopfield network, $w_{ii} = 0$, the off-diagonal entries are symmetrical, *i.e.*, $w_{ij} = w_{ji}$, for all i and j, and the network outputs are discrete (Hopfield 1982, 1984). Recently, more general Hopfield network formulations with $w_{ii} \neq 0$ and continuous-valued outputs have been developed (Hopfield 1984; Hopfield and Tank 1985). At iteration k, the input to neuron i is obtained by:

$$net_i^{(k)} = \sum_{j=1}^{n} w_{ij} v_j^{(k-1)}. \tag{3.1}$$

Each neuron has a transfer (activation) function which converts the neuron's input $net_i^{(k)}$ to output $v_i^{(k)}$. Initially, Hopfield proposed the use of the McCulloch-Pitts neurons (Hopfield 1982), also called threshold logic neurons. In these neurons, the synaptic input $net_i^{(k)}$ in (3.1) is compared with the neuron's threshold θ_i and transformed to an output of the form (3.2).

$$v_i^{(k)} = \begin{cases} 0, & \text{if } net_i^{(k)} < \theta_i \\ 1, & \text{if } net_i^{(k)} > \theta_i \\ v_i^{(k-1)}, & \text{if } net_i^{(k)} = \theta_i \end{cases}. \tag{3.2}$$

More recently, Hopfield (1984) proposed the use of a continuous-valued sigmoidal activation function of the type in (3.3), where β controls the slope of the function. Often, $T = 1/\beta$ is called the temperature of the function. Of course, other types of activation functions are possible as well.

$$v_i^{(k)} = \left(1 + e^{-\beta net_i^{(k)}}\right)^{-1}. \tag{3.3}$$

The output of the network at one iteration serves as input at the next iteration. This process is repeated dynamically until either the network converges to a stable state or a pre-specified number of iterations has been performed. In the latter case, the network may be unstable. The dynamic

convergence to a stable state minimizes an "energy function" which measures the overall difference between the network outputs of consecutive iterations. A network converges to a stable state when it reaches a local or global minimum energy point. For the mathematical details we refer the readers to Hopfield (1982, 1984).

3.2. A Modified Hopfield Network for Determining the AHP Preference Ratings

We will next show how a Hopfield network formulation with a customized activation function can be used to compute the principal eigenvector of the reciprocal comparison matrix, *i.e.* the preference weights in the AHP, and how the dynamics of this Hopfield network formulation are equivalent to the power method, as long as the $n \times n$ matrix $\mathbf{A}$ of pairwise preference judgments is fully specified.

In contrast with the symmetric weights of the original Hopfield network, this modified Hopfield network formulation has reciprocal weights ($w_{ij} = a_{ij} = 1/w_{ji} = 1/a_{ji}$), so that the connectivity weight matrix $\mathbf{W} = \mathbf{A}$ is asymmetric. Moreover, we set $w_{ii} = 1$. The output values of the network at each iteration represent the estimates of the principal eigenvector components. Since $\mathbf{A}$ is positive irreducible, the Hopfield network is the irreducible graph of $\mathbf{A}^{\mathrm{T}}$. Rather than selecting activation functions that have traditionally been used for Hopfield networks, such as threshold and sigmoidal functions in (3.2) and (3.3), we customize the transformation of the inputs $net_i^{(k)}$ to outputs $v_i^{(k)}$, according to (2.5). Hence, the input to the neurons is defined by $\boldsymbol{net}^{(k)} = \mathbf{A}\mathbf{v}^{(k-1)}$, which corresponds exactly to equation (2.4) of the power method, and the neuron output equals $\mathbf{v}^{(k)}$, normalized using either the L_1- or L_∞-norm as in (2.5).

Since each step of the power method is identical to one iteration of this modified Hopfield network formulation, the computational aspects of the two methods are equivalent. Thus, the modified Hopfield network with reciprocal connectivity weights and the node activation functions in (2.5) will converge to a stable (optimal) state which is identical to the solution obtained using the power method, and the conditions under which both methods will converge to a finite nonzero solution are the same. While the power method converges to the principal eigenvector, the Hopfield network memorizes *exactly one vector*, the principal eigenvector of the connectivity matrix.

4. FEED–FORWARD NEURAL NETWORK FORMULATION FOR THE AHP

4.1. Feed-Forward *ANN*

A feed-forward *ANN* is an *ANN* where the nodes are organized into layers, and the directed and weighted arcs from a node are only linked to nodes in the next higher layer (Rumelhart, Hinton and Williams 1986; Wasserman 1989). Nodes in the input layer accept input from the outside world and nodes in the output layer generate output to the outside world. The input nodes are used to distribute inputs only, and do not serve any processing or computational function. The output of the output layer is the output of the feed-forward *ANN*. Nodes in layers between the input layer and the output layer are called hidden nodes, and the corresponding layers are the hidden layers. Hidden layers enable a feed-forward *ANN* to represent complex mappings.

Denote the connectivity weight from node r in layer $i-1$ to node k in layer i by w_{kr}^{i}. Each node, except for the input nodes, has a node bias or threshold. Denote the node bias of node k in layer i by θ_{k}^{i}. Further, denote the set of connectivity weights and node biases by $W = \{w_{kr}^{i},\ \theta_{k}^{i}\}$. The components of W are usually determined throughout the training process.

A feed-forward *ANN* maps from the input space $\Re^{n_I}$ to the output space $\Re^{n_O}$, that is, for any given input vector $\mathbf{a} \in \Re^{n_I}$, the network computes an output vector $\mathbf{o} \in \Re^{n_O}$. The mapping is a dynamic process, in which node inputs and outputs are computed sequentially from the input layer to the output layer. For $i > 0$, the input to node k in layer i, denoted by net_{k}^{i}, is the weighted sum of the outputs of all nodes directly connected to it in layer $i-1$ plus θ_{k}^{i}, *i.e.*,

$$net_{k}^{i} = \sum_{r=1}^{n_{i-1}} w_{kr}^{i} o_{r}^{i-1} + \theta_{k}^{i}, \tag{4.1}$$

where o_{r}^{i-1} is the output of node r in layer $i-1$, and n_{i-1} is the number of nodes in layer $i-1$. Each node has an activation function which computes the node's output based on its input. The most frequently used activation function is the sigmoidal function, as in (4.2),

$$o_{k}^{i} = \left(1 + e^{-\beta net_{k}^{i}}\right)^{-1}. \tag{4.2}$$

Feed-forward *ANN*s have been applied to many non-trivial numerical tasks (Lippmann 1987) and real world problems, notably to classification and pattern recognition problems (Másson and Wang 1990; Zahedi 1991). Recently, feed-forward *ANN*s have also been proposed for solving multicriteria decision problems (Wang and Malakooti 1992; Malakooti and Zhou 1994). Sanger (1989) proposed an unsupervised learning algorithm in a single layer feed-forward *ANN* that is similar conceptually to algorithms in principal components analysis and factor analysis. The mathematics of these statistical techniques are similar conceptually to those underlying the approximation of the principal eigenvector.

4.2. The Feed-Forward *ANN* Formulation and Training

A feed-forward *ANN* is usually trained to represent an unknown mapping. The purpose of training is to determine the values of the components of W such that the network closely represents the unknown mapping. In the training process, the network learns from training patterns in the training set, which is a collection of paired input and output vectors observed from the unknown mapping. Once a feed-forward *ANN* has been trained, the knowledge of the unknown mapping is stored in the components of W.

The training of a feed-forward *ANN* is accomplished by applying input vectors from the training set to the network, mapping the input vectors to the computed output vectors, and then comparing the computed output vectors with the desired or target output vectors in the training set, after which the values of the components of W are adjusted to reduce the differences between the computed and desired output vectors. After a number of training iterations, the components of W will converge to a set of values that minimize the difference between the computed and desired output vectors, and the network will organize itself internally, constructing a model to represent the unknown mapping from $\Re^{n_I}$ to $\Re^{n_O}$. A well-trained *ANN* is able to generalize to patterns it has never seen before. Any new input vector presented to an appropriately trained network will yield an output similar to the one which would have been given by the actual mapping.

Given their generalizing capability, feed-forward *ANN*s are used to approximate the mapping from a reciprocal comparison matrix in the AHP to the associated preference ratings of the DM. The preference information provided by the DM through the $n \times n$ pairwise comparison matrix $\mathbf{A}$ serves as the input to the feed-forward neural network. Let $\mathbf{a} \in \Re^{n(n-1)/2}$ be the vector consisting of the elements of $\mathbf{A}$ above the main diagonal, *i.e.*, those a_{ij} for which $i < j$, ordered such that $\mathbf{a} = (a_{12}, \ldots, a_{1n}, a_{23}, \ldots, a_{2n}, \ldots, a_{n-1,n})^{\mathrm{T}}$. Since $\mathbf{A}$ is reciprocal, $\mathbf{a}$ contains all relevant preference information in $\mathbf{A}$, and is used as the input to the feed-forward *ANN*. Thus, the neural network will have $n_I = n(n-1)/2$ input nodes. The desired output vector consists of the n-dimensional true preference rating vector $\mathbf{r}$ of the DM, so that the number of output nodes is $n_O = n$. The compound vector $(\mathbf{a}^{\mathrm{T}}, \mathbf{r}^{\mathrm{T}})$ represents one training pattern in the training set. The number of hidden layers and the number of hidden nodes in each hidden layer depends on the complexity of the mapping. In this paper, we limit ourselves to pairwise comparison matrices of sizes 3×3, 4×4, 5×5 and 6×6.

The software NeuralWorks Professional II/Plus (NeuralWare 1993a) was used to train separate neural nets for each matrix size. For the problems analyzed in this paper, networks with two hidden layers provided good results. After experimenting with a number of alternative network configurations, training set sizes and network parameter settings, we found the network configurations in Table 1 to

yield tolerable errors and to converge within reasonable time. The number of training patterns P in the training set used for these different matrix sizes is given in Table 1. During the training process, the learning rates were gradually decreased from between 0.8–0.9 to 0.1, and the momentum factors from 0.6–0.7 to 0.05. Other network parameters not mentioned in Table 1 were kept at the default values of the NeuralWorks software package. Larger problems (5×5 and 6×6) are more complicated than smaller ones (3×3 and 4×4), and consequently require more nodes in the hidden layers and more training patterns in the training set to represent the mapping. The network in Figure 1 is a graphical representation of a feed-forward *ANN* for the 3×3 pairwise comparison matrices, with 10 nodes in the first hidden layer and 6 in the second.

Figure 1 and Table 1 About Here

In this study, the components of the network output vector $\mathbf{o} = (o_1, \ldots, o_n)^T$ are normalized to sum to unity. Denoting training pattern t by $(\mathbf{a}_t^T, \mathbf{r}_t^T)$, and the corresponding normalized output of the *ANN* by $\mathbf{o}_t$, the network error associated with training pattern t is measured by the root mean square error (RMSE_t) (NeuralWare 1993b) given in (4.3):

$$\mathrm{RMSE}_t = \sqrt{\frac{1}{n_O} \sum_{j=1}^{n_O} (o_{tj} - r_{tj})^2} \,. \tag{4.3}$$

The network error associated with all P patterns in the training set is measured by $\overline{\mathrm{RMSE}}$ as in (4.4).

$$\overline{\mathrm{RMSE}} = \frac{1}{P} \sum_{t=1}^{P} \mathrm{RMSE}_t \tag{4.4}$$

In the training process, $\overline{\mathrm{RMSE}}$ is minimized. In order to avoid pattern memorization or network paralysis due to overtraining, throughout the training process the progress of training was monitored by calculating $\overline{\mathrm{RMSE}}$ for 200 randomly generated verification patterns that were *not* part of the training set. Each network was trained either until $\overline{\mathrm{RMSE}} < 0.001$ for these 200 patterns in the verification set, or until it appeared that the $\overline{\mathrm{RMSE}}$ of the training set could not be improved further. The computational effort of training the *ANN*s is considerable, but not prohibitive. A typical training session using the NeuralWare software package (1993a, 1993b) on a compatible 486 PC, 25 MHz, took several hours. This computational time is reasonable, since only one network needs to be trained for each matrix size. Once trained, this *ANN* can be used to predict the ratings for *any* pairwise comparison matrix of that size.

5. EXPERIMENTAL RESULTS

At this point, we need to distinguish between two issues: (1) whether a feed-forward *ANN* is able to closely approximate the mapping from a pairwise comparison matrix to its principal eigenvector, and (2) whether a feed-forward *ANN* representation is more accurate than the power method in estimating the preference ratings if the preference information is imprecise. The former issue is addressed in Section 5.1, while the latter is analyzed in Section 5.2.

5.1. Approximation of the Principal Eigenvector Using Feed-Forward Neural Networks

In this section, we only compare the output vector of a trained *ANN* with the actual principal eigenvector obtained with the power method, and do not consider the issue of imprecise preference judgments. In order to verify that feed-forward *ANN*s are indeed capable of yielding close approximations of the principal eigenvectors, we trained separate networks for two separate types of problem settings: (1) the pairwise comparison matrix **A** of each training pattern is *perfectly consistent*, and (2) **A** is *moderately inconsistent* (Saaty 1988). Thus, 8 different neural networks were trained for the purpose of the analysis in this section (4 problem sizes and 2 problem settings). As the goal is to verify if the output of the trained network can approximate the principal eigenvector of **A**, we used the normalized principal eigenvector of **A** (either consistent or inconsistent) obtained with the power method as the desired output in the training process.

Saaty (1988) argues that few DMs are fully consistent in their pairwise preference judgments, and inconsistency indices of less than 0.1 are reasonable in practice. Accordingly, we randomly generated the matrices of inconsistent judgments such that their inconsistency index did not exceed 0.1. We decided against including matrices with inconsistency indices exceeding 0.1, as in such cases there may be a fundamental flaw in the elicited preference information, in which case Saaty (1988) recommends re-evaluating and double-checking the judgments, rather than proceeding with the preference analysis.

Given the desired output vector $\mathbf{r}_t$ and the predicted output vector $\mathbf{o}_t$ of pattern t, the angle γ_t (in degrees) between $\mathbf{r}_t$ and $\mathbf{o}_t$ is determined by (5.1) and the maximum bit error (MBE_t), *i.e.* the largest absolute error among all output nodes, is defined in (5.2):

$$\gamma_t = \frac{360}{2\pi} \arccos \frac{\mathbf{o}_t \cdot \mathbf{r}_t}{|\mathbf{o}_t|\,|\mathbf{r}_t|} = \frac{360}{2\pi} \arccos \frac{\sum_{j=1}^{n} o_{tj} r_{tj}}{\sqrt{\sum_{j=1}^{n} o_{tj}^2}\ \sqrt{\sum_{j=1}^{n} r_{tj}^2}}, \tag{5.1}$$

$$\mathrm{MBE}_t = \max_j \left\{ |\, o_{tj} - r_{tj} \,| \right\}. \tag{5.2}$$

In network training, the $\overline{\text{RMSE}}$ of (4.3) was used as a measure of network error. In model validation, we used not only $\overline{\text{RMSE}}$, but also two alternative error measures, the average angle $\overline{\gamma}$ and the average maximum bit error $\overline{\text{MBE}}$ of all validation patterns, as defined in (5.3) and (5.4), respectively:

$$\overline{\gamma} = \frac{1}{P} \sum_{t=1}^{P} \gamma_t, \tag{5.3}$$

$$\overline{\text{MBE}} = \frac{1}{P} \sum_{t=1}^{P} \text{MBE}_t. \tag{5.4}$$

We use the symbol P both for the size of the training and validation sets, but this should not cause any confusion. Obviously, smaller values of $\overline{\gamma}$ and $\overline{\text{MBE}}$ (and of $\overline{\text{RMSE}}$, of course) are indicative of higher overall degrees of similarity, and thus of higher predictive accuracy of the *ANN* models. Smaller values of the $\overline{\text{MBE}}$ are indicative of a more favorable worst case accuracy of the neural network.

Tables 2 and 3 About Here

Tables 2 and 3 summarize the predictive abilities ($\overline{\text{RMSE}}$, $\overline{\text{MBE}}$ and $\overline{\gamma}$) of the 8 trained *ANN*s, comparing the actual principal eigenvectors with the output vectors of the trained *ANN*s for 1,000 independently generated moderately inconsistent and perfectly consistent validation patterns respectively that were not in the training set. From Table 2, we see that the values of $\overline{\text{RMSE}}$, $\overline{\text{MBE}}$ and $\overline{\gamma}$ for the *ANN*s trained using inconsistent preference information are uniformly lower than the corresponding values for the *ANN*s trained using consistent preference information. For instance, the *ANN* for the 4×4 matrices trained with perfect consistency has a $\overline{\text{RMSE}}$ of 0.054, a $\overline{\text{MBE}}$ of 0.083 and a $\overline{\gamma}$ of 9.728, whereas the *ANN* trained using inconsistent 4×4 matrices yields values of 0.010, 0.010 and 1.919, respectively. This suggests, as one might expect, that it is better to predict the preference ratings based on moderately inconsistent pairwise comparison matrix using an *ANN* that was *also trained* on moderately inconsistent preference information, rather than using an *ANN* trained on perfectly consistent preferences. From Table 3 it is clear that, as anticipated, those *ANN*s that were trained on consistent data generally predict rating vectors of consistent pairwise comparison matrices more accurately than those trained on inconsistent data. For example, for the *ANN* for the 6×6 matrices trained with perfect consistency, the values of $\overline{\text{RMSE}}$, $\overline{\text{MBE}}$ and $\overline{\gamma}$ are 0.004, 0.007 and 0.913, respectively, while the corresponding values for the *ANN* trained using moderately inconsistent matrices are 0.010, 0.018 and 2.840.

Summarizing, Tables 2 and 3 confirm the notion that the predictive accuracy of the *ANN* will be higher as the characteristics of the training and validation sets match more closely. Importantly, the low values of $\overline{\text{RMSE}}$, $\overline{\text{MBE}}$ and $\bar{\gamma}$ in both Table 2 and Table 3 clearly indicate that the trained *ANN*s are capable of closely approximating the principal eigenvectors of the pairwise comparison matrices, no matter whether inconsistency is involved.

5.2. Network Generalization: Predicted Ratings for Imprecise Preference Structures

In this section, we investigate further the generalization properties and robustness of the feed-forward *ANN*s by comparing their predictive accuracy with that of the power method, in the presence of imprecise preference judgments. In the simulation experiments below, we perturbed the components of the perfectly consistent pairwise comparison matrices to introduce impreciseness into the preference judgments. The purpose of introducing impreciseness is to simulate actual decision making situations. Although there are many ways to introduce impreciseness in a systematic simulation experiment, none of which will correspond exactly with an actual decision situation, the limited simulation discussed in this section appears sufficient for the purpose of assessing the viability of estimating preference ratings using *ANN* models. However, this study does not intend to provide a comprehensive investigation of the conditions under which such models perform best.

For the purpose of this study, we assume that the DM's *true* preference structure is perfectly *consistent* and *known*, but the *elicited* pairwise comparison matrix is moderately *inconsistent*, reflecting the fact that the DM may provide somewhat imprecise preference information. Accordingly, we constructed the training set in the following way. First we generated perfectly consistent pairwise comparison matrices **A** with normalized principal eigenvectors **r**, and then perturbed the components of **A** to obtain moderately inconsistent matrices **B** according to (5.5),

$$b_{ij} = a_{ij} + \eta U a_{ij}, \tag{5.5}$$

where U is a uniformly distributed random variate over the interval $[-1, +1]$ and η is a constant. Thus, the mean disturbance factor $\mu_{\eta U a_{ij}}$ equals 0, with a standard deviation of $\sigma_{\eta U a_{ij}} = 0.57735 \eta a_{ij}$. All matrices in the training set were generated using $\eta = 0.8$, so that $\sigma_{\eta U a_{ij}} = 0.46188 a_{ij}$, implying a mild perturbation. In this study, the average inconsistency indices of the perturbed matrices in the training sets were between 0.02 (for the 3×3 matrices) and 0.06 (for the 6×6 matrices). Any perturbed matrix with an inconsistency index exceeding 0.1 was discarded. We ensured that all elements of **B** were within the range of 1/9 to 9. Also, we ensured that the direction of the individual relative preferences was not reversed during the perturbation process (*i.e.*, $b_{ij} > 1$ if $a_{ij} > 1$, and $b_{ij} < 1$ if $a_{ij} < 1$). Let **b** represent the vector consisting of the components b_{ij} of **B** with $i < j$, with the

components ordered in the same way as in **a**, so that the compound vector ($\mathbf{b}^T$, $\mathbf{r}^T$) is a training pattern in the training set. Other than the fact that the preference patterns in the training set now reflect impreciseness, the network training was conducted in the way described in Section 4, using the network configurations and number of training and verification patterns as specified in Table 1. By doing so, we enabled the *ANN* to learn how to deal with somewhat imprecise preference judgments.

Table 4 About Here

Two types of validation samples were used in model validation, each consisting of 1,000 independent replications. None of the validation patterns is part of the training set. First, we generated each perturbed pairwise comparison matrix **C** in exactly the same way as we did for **B**, again with $\eta = 0.8$, *i.e.*, $c_{ij} = a_{ij} + \eta U a_{ij}$. We experimented with other values of η, and obtained similar results to those reported in Table 4. Table 4 shows that the average inconsistency index of the matrices **C** for the different matrix sizes varies between 0.025 and 0.055.

Realizing that the way in which the **C** were generated might bias the performance in favor of the *ANN* models, we generated doubly perturbed validation matrices **D**, by first generating perturbed matrices in the same way as **C** above, with $\eta = 0.8$, and then perturbing these matrices once again with a factor $\eta = 0.8$, so that $d_{ij} = c_{ij} + \eta U c_{ij}$. Again, in generating these matrices, any matrix with an inconsistency index exceeding 0.1, with elements outside the range of [1/9, 9], or with reversals in the preference direction was excluded from the analysis. From Table 4 we see that, depending on the matrix size, the average inconsistency of the **D** matrices varies between 0.058 and 0.071. While of course higher than those of **C**, these figures are well below 0.1. As the *ANN*s had not been trained directly to recognize the kind of impreciseness created by the double perturbation, the matrices **D** provide an additional means of testing the generalizing abilities of the *ANN* models.

Since the vector of actual preference ratings, *i.e.* the principal eigenvector of **A**, for each validation pattern is known, we can compute $\overline{\text{RMSE}}$, $\overline{\text{MBE}}$ and $\overline{\gamma}$ separately for the *ANN* models and for the power method and use them to measure the relative performance of the two methods. In order to support the conclusions, we used the validation samples to test the null hypothesis that a given *ANN* model did not yield more accurate predicted ratings than the power method, on average, against the alternative that it did provide more accurate results, on average. The T-test statistics of difference in means for all three performance measures ($\overline{\text{RMSE}}$, $\overline{\text{MBE}}$ and $\overline{\gamma}$) are provided in Table 4.

The results in Table 4 show that without exception the feed-forward *ANN*s predict the preference ratings more accurately than the power method, since all t-values are negative. In each case, the difference in mean accuracy is significant at the $\alpha = 0.05$ level. All except one t-values are in fact

significant at the $\alpha = 0.01$ level. These results hold for all three performance measures, and for the once-perturbed, C, as well as for the twice-perturbed, D, matrices. In fact, the t-values associated with the once- and twice-perturbed matrices were remarkably similar.

Even though each trained network converged and proved quite accurate and robust in predicting the preference ratings, *ANN* representations of these mappings are not unique, so that it might be possible to obtain results similar to or better than those reported in this paper when alternative network configurations are used. Nevertheless, we believe that the robustness of the particular networks reported in this section clearly illustrates the effectiveness of using feed-forward *ANN* for modeling AHP-type preference structures, and more accurate *ANN* architectures would only strengthen the conclusions of this study.

The conclusion from this limited simulation experiment is that – at least for the type of problems used in this study – *ANN* representations of ratio-scale pairwise preference structures can provide robust estimates of the DM's preference ratings. This study clearly provides preliminary but compelling results. Further studies evaluating various different ways in which the impreciseness phenomenon can play a role in a DM's assessment of relative preferences should also provide a more definite answer to the question to which extent *ANN* representations of preference structures are viable.

The feed-forward *ANN* representations can easily be included in any AHP software package, just as the power method for calculating the principal eigenvector. Since a given feed-forward *ANN* is trained to learn and generalize the mapping from *any* positive $n \times n$ reciprocal comparison matrix to its preference rating vector, it can be used to approximate the preference ratings for any decision situation involving n alternatives. Hence, implementation of the proposed *ANN*s merely involves embedding the already trained *ANN*s in the existing software. The user would then have the option to use the *ANN* approach (rather than the power method) if the pairwise comparison values elicited from a given DM are likely to be imprecise.

6. CONCLUSIONS

There is ample evidence that certain *ANN* structures are well-suited for accurately identifying patterns, if these patterns are imprecise or fuzzy (Kohonen 1984; Kosko 1990; Pao 1989; Wasserman 1989). In the AHP, the patterns consist of the DM's preference statements in the form of pairwise comparison matrices. In this paper, we investigate two different *ANN* structures in terms of accurately approximating (generalizing) the preference ratings of the alternatives, if some or all of the preference judgments are imprecise.

The dynamics of the modified Hopfield network formulation correspond exactly with the power method, but this formulation is inflexible and unable to generalize the resulting ratings. Hence, even

though the connection between the modified Hopfield network and the AHP is interesting in itself, this network formulation appears to be of limited use in practice. However, the feed-forward *ANN* formulation is approximate and is able to generalize the preference structures, yielding robust preference ratings when the preference statements are imprecise.

Several researchers have recently studied the problem of how the preference ratings of the alternatives can be approximated if the preference information is incomplete or imprecise (Harker 1987; Salo and Hämäläinen 1991, 1992, 1994; Wedley 1993; Wedley, Schoner and Tang 1993). This is an important research topic, as in many real life decision problems the preference information elicited from the DM is imprecise. In the current study, we have found preliminary but compelling evidence that, due to their generalizing abilities, *ANN* representations can provide robust approximations of a DM's preferences when the preference information at hand is imprecise. Hence, it appears that the feed-forward *ANN* formulation, perhaps in conjunction with methodologies aimed at reducing the degree of impreciseness during the preference elicitation process (Salo and Hämäläinen 1991, 1992, 1994), is potentially a powerful tool in determining preference ratings for effective decision support.

The *ANN* representation appears even more attractive, because once one network has been trained for each matrix size, the trained *ANN*s can easily be embedded in any AHP software package. Thus the user does not need to be familiar with *ANN*s, and is not required to train a network for his/her particular application.

The analysis in this paper is based, directly or indirectly, on the methodology of the conventional AHP (Saaty 1988). As alluded to in Section 2, there exist many worthwhile alternative ways to analyze and interpret the information contained in pairwise comparison matrices. Most of these variants of the AHP (as well as perhaps other preference representation, such as value functions) can easily be implemented within an *ANN* framework. Moreover, impreciseness can be operationalized and introduced to the analysis in numerous different ways, and can take on many different characteristics that depend on the particular decision situation. It may also be useful to explore *ANN* paradigms other than Hopfield and feed-forward networks. It appears that extending the current study into these research directions will provide interesting and useful findings, both for the theory and practice of decision analysis.

REFERENCES

Barzilai, J., W. D. Cook and B. Golani, "Consistent Weights for Judgment Matrices of the Relative Importance of Alternatives," *Operations Research Letters*, **6**, 131–141, 1987.

Burden, R. L., and J. D. Faires, *Numerical Analysis*, Fourth Edition, PWS-KENT, Boston, MA, 1989.

Cohen, A. M., J. F. Cutts, R. Fielder, D. E. Jones, J. Ribbans, and E. Stuart, *Numerical Analysis*, Wiley, New York, NY, 1973.

Dyer, J. S., "Remarks on the Analytic Hierarchy Process," *Management Science*, **36**, 249–258, 1990.

Harker, P. T., "Incomplete Pairwise Comparisons in the Analytic Hierarchy Process," *Mathematical Modeling*, **9**, 837–848, 1987.

Hopfield, J. J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", in *Proceedings of the National Academy of Science*, **79**, 2554–2558, 1982.

Hopfield, J. J., "Neurons with Graded Response Have Collective Computational Properties like Those of Two-State Neurons", in: *Proceedings of the National Academy of Science*, **81**, 3088–3092, 1984.

Hopfield, J. J. and D. W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, **52**, 141–152, 1985.

Kohonen, T., *Self Organization and Associative Memory*, Series in Information Sciences, **8**, Springer Verlag, New York, NY, 1984.

Kosko, B., *Neural Networks and Fuzzy Systems*, Prentice Hall, Englewood Cliffs, NJ, 1990.

Lippmann, R. P., "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, **4**, 4–22, 1987.

Másson, E. and Y. J. Wang, "Introduction to Computation and Learning in Artificial Neural Networks," *European Journal of Operational Research*, **47**, 1–28, 1990.

Malakooti, B. and Y. Zhou, "An Adaptive Feedforward Artificial Neural Network with Application to Multiple Criteria Decision Making," *Management Science*, Forthcoming, 1994.

NeuralWare, *NeuralWorks Professional II/Plus: Reference Guide*, NeuralWare, Inc., Pittsburgh, PA, 1993a.

NeuralWare, *Using NeuralWorks: A Tutorial for NeuralWorks Professional II/Plus and NeuralWorks Explorer*, NeuralWare, Inc., Pittsburgh, PA, 1993b.

Pao, Y. H., *Adaptive Pattern Recognition and Neural Networks*, Addison Wesley, Reading, MA, 1989.

Pöyhönen, M. A., R. P. Hämäläinen and A. A. Salo, "An Experiment on the Numerical Modeling of Verbal Ratio Statements," Research Report A50, Systems Analysis Laboratory, Helsinki University of Technology, 1994.

Rumelhart, D. E., G. E. Hinton and R. J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing, Volume 1: Foundations*, D. E. Rumelhart and J. L. McClelland (Eds.), MIT Press, Cambridge, MA, 318–362, 1986.

Saaty, T. L., "A Scaling Method for Priorities in Hierarchical Structures," *Journal of Mathematical Psychology*, **15**, 237–281, 1977.

Saaty, T. L., *Multicriteria Decision Making: The Analytic Hierarchy Process (Revised Edition)*, University of Pittsburgh, 1988.

Salo, A. A. and R. P. Hämäläinen, "Ambiguous Preference Statements in the Analytic Hierarchy Process," Research Report A39, Systems Analysis Laboratory, Helsinki University of Technology, 1991.

Salo, A. A. and R. P. Hämäläinen, "Preference Assessment by Imprecise Ratio Statements," *Operations Research*, **40**, 1053–1061, 1992.

Salo, A. A. and R. P. Hämäläinen, "Preference Programming Through Approximate Ratio Comparisons," *European Journal of Operational Research*, Forthcoming, 1994.

Sanger, T. D., "Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network," *Neural Networks*, **2**, 459–473, 1989.

Schoner, B. and W. C. Wedley, "Ambiguous Criteria Weights in AHP: Consequences and Solutions," *Decision Sciences*, **20**, 462–475, 1989.

Wang, J. and B. Malakooti, "A Feedforward Neural Network for Multiple Criteria Decision Making," *Computers and Operations Research*, **19** (2), 151–167, 1992.

Wasserman, P. D., *Neural Computing, Theory and Practice*, Van Nostrand Reinhold, New York, NY, 1989.

Wedley, W. C., "Consistency Prediction for Incomplete AHP Matrices," *Mathematical and Computer Modelling*, **17** (4/5), 151–161, 1993.

Wedley, W. C., B. Schoner and T. S. Tang, "Starting rules for Incomplete Comparisons in the Analytical Hierarchy Process," *Mathematical and Computer Modelling*, **17** (4/5), 93–100, 1993.

Winkler, R. L., "Decision Modeling and Rational Choice: AHP and Utility Theory," *Management Science*, **36**, 247–248, 1990.

Zahedi, F., "An Introduction to Neural Networks and a Comparison with Artificial Intelligence and Expert Systems," *Interfaces*, **21** (2), 25–38, 1991.

Figure 1: Example of a Feed-Forward Neural Network Architecture for Pairwise Comparison Matrices of Size 3×3

Table 1: Network Configurations for Various Size Pairwise Comparison Matrices

Matrix Size ($n \times n$)	Size of the Training Set (P)	Number of Training Epochs Cons.	Incons.	Size of the Verification Set[1]	Size of the Validation Sample[2]	Number of Hidden Layers	Number of Hidden Nodes[3]
3×3	1,000	1,200	2,400	200	1,000	2	10, 6
4×4	1,000	1,200	2,400	200	1,000	2	20, 8
5×5	3,000	490	800	200	1,000	2	25, 8
6×6	3,000	490	800	200	1,000	2	30, 10

1: The verification patterns are not part of the training set, and serve as an unbiased monitor of the training process.
2: The validation patterns serve as an unbiased ex-post validation sample for measuring network accuracy.
3: These figures represent the number of hidden nodes in the first and second hidden layer, respectively.

Table 2: Prediction Accuracy of Networks for Moderately Inconsistent validation Samples[1]

Matrix Size	Perfectly Consistent Training Sets			Moderately Inconsistent Training Sets[1]		
	$\overline{\gamma}$ (Std.)[2]	$\overline{\text{MBE}}$ (Std.)	$\overline{\text{RMSE}}$ (Std.)	$\overline{\gamma}$ (Std.)	$\overline{\text{MBE}}$ (Std.)	$\overline{\text{RMSE}}$ (Std.)
3×3	7.976 (5.539)	0.078 (0.051)	0.059 (0.039)	1.103 (0.646)	0.010 (0.006)	0.008 (0.005)
4×4	9.728 (5.686)	0.083 (0.050)	0.054 (0.031)	1.919 (1.151)	0.010 (0.001)	0.010 (0.006)
5×5	11.759 (6.323)	0.088 (0.050)	0.051 (0.028)	2.332 (1.198)	0.017 (0.009)	0.010 (0.005)
6×6	13.081 (5.792)	0.091 (0.044)	0.048 (0.021)	2.906 (1.292)	0.019 (0.009)	0.011 (0.005)

1: All of the training patterns have an inconsistency ratio (Saaty 1988) between 0 and 0.1.
2: "Std." stands for standard deviation.

Table 3: Prediction Accuracy of Networks for Perfectly Consistent Validation Samples

Matrix Size	Perfectly Consistent Training Sets			Moderately Inconsistent Training Sets[1]		
	$\overline{\gamma}$ (Std.)[2]	$\overline{\text{MBE}}$ (Std.)	$\overline{\text{RMSE}}$ (Std.)	$\overline{\gamma}$ (Std.)	$\overline{\text{MBE}}$ (Std.)	$\overline{\text{RMSE}}$ (Std.)
3×3	0.435 (0.324)	0.005 (0.004)	0.003 (0.003)	0.878 (0.592)	0.009 (0.006)	0.007 (0.004)
4×4	0.428 (0.301)	0.004 (0.003)	0.002 (0.002)	1.759 (1.084)	0.015 (0.009)	0.010 (0.006)
5×5	0.924 (0.608)	0.007 (0.005)	0.004 (0.003)	2.349 (1.232)	0.017 (0.010)	0.010 (0.005)
6×6	0.913 (0.578)	0.007 (0.004)	0.004 (0.002)	2.840 (1.211)	0.018 (0.008)	0.010 (0.004)

1: All of the training patterns have an inconsistency ratio (Saaty 1988) between 0 and 0.1.
2: "Std." stands for standard deviation.

Table 4: Tests of Significant Difference in Mean Predictive Accuracy Between Neural Network Models and Power Method, for Perturbed Preference Matrices[1]

Matrix Size	Average Inconsistency Index		Once-Perturbed Matrices (C)			Twice-Perturbed Matrices (D)		
			t-Values for			t-Values for		
	C	D	$\bar{\gamma}$	$\overline{\text{MBE}}$	$\overline{\text{RMSE}}$	$\bar{\gamma}$	$\overline{\text{MBE}}$	$\overline{\text{RMSE}}$
3×3	0.025	0.058	−5.65	−5.88	−5.58	−4.83	−5.07	−4.70
4×4	0.041	0.071	−2.26	−2.76	−2.48	−3.00	−3.32	−3.16
5×5	0.050	0.071	−8.49	−8.52	−8.59	−9.50	−9.26	−9.58
6×6	0.055	0.071	−7.77	−6.62	−7.38	−6.53	−5.18	−6.18

1: Null hypothesis: the *ANN* model does not predict the preference weights more accurately than the power method; Alternative hypothesis: the *ANN* model does predict more accurately. t-values below -1.67 and -2.38 indicate that the null hypothesis is rejected, thus implying that the neural network predictions are significantly better at the $\alpha = 0.05$ and $\alpha = 0.01$ level, respectively.