USING NEURAL NETWORKS FOR PRIORITY DERIVATION WITH INTERVAL JUDGMENTS

Gang Hao
Luis G. Vargas
Joseph M. Katz Graduate School of Business
University of Pittsburgh
Pittsburgh, Pa. 15260

ABSTRACT

Interval judgments is a natural and easy way to encode uncertainty in pairwise comparisons. The major issue with interval judgments is how to derive priorities from the interval statements, which captures decision makers' preferences. There have been two types of approaches to deal with the problem: mathematical programming based methods and probabilistic methods. This paper introduces a new approach --- a Neural Networks based method. We formulate the process of interval judgments as a Neural Network model and then explore two ways of deriving the priorities using a revised Backpropagation algorithm. The proposed method shows some potential advantages in the sense of less computation and a simpler mechanism in the modeling of the problem. The method is illustrated with numerical examples. The results of some comparative analysis with other current available approaches are also included.

1. Introduction

The basic purpose of decision analysis is to provide decision aids, such as a models, methods, techniques or processes to help decision makers to recommend actions. Ranking decision alternatives by means of pairwise comparisons according to criteria is one of the commonly used methods for such purpose. There are two basic issues that must be considered in this method. First, how to make and represent pairwise comparisons, specially when the criteria are of a qualitative nature. And second, how to extract the ranking form the pairwise comparison statements.

To compare object A with object B, the decision maker needs to answer the following questions: Given a particular shared criterion or property, which of the two objects is more important according to this criterion, and how much more important is it? Paired comparisons are particularly useful when there exist no judgmental standards to refer to. However, since the human mind has limited capabilities of information handling, we are always faced with incomplete knowledge, imperfect information, and an inadequate version of the true states of events. For instance, if we compare student A with student B with respect to their intelligence, or car A with car B according to their quality, it is usually difficult to tell exactly how much intelligent one student is than another or how much better one car is than another. We may be unsure of the true states of the objects we are comparing; we may not feel confident on our judgments, or we may be ambiguous about the "right" expression of our judgments. We need to find ways to elicit such comparisons with less requirements to decision makers, and encode the results with satisfactory accuracy, especially when faced with uncertainties.

Our objective is to find ways of simplifying the pairwise comparisons process and develop algorithms the produce satisfactory priorities. These two issues are also primary focus of this paper. In addition to some general discussion about these basic issues, the particular method of pairwise comparison we are interested in here is the methods of interval judgments and the typical algorithm for priority derivation we are going to propose is based on an Artificial Neural Networks.

Our paper is organized in following format. Section 2 provides some general discussions about coping with uncertainty in pairwise comparisons. Section 3 gives the structure of interval judgments and also briefly reviews the current available interval judgments approaches. In Section 4, we include a brief introduction of Neural Networks and define the terminologies relevant to our later discussions. In Section 5, we will show how to formulate the interval judgment problem as a Neural Network model and propose two ways of using a revised Backpropagation algorithm for priority derivation. Section 6 and 7 address some conclusions of our study including a brief discussion of the comparison of the proposed algorithm with other available methods and lay out issues for future research.

2. Encoding the uncertainty in pairwise comparisons

To capture the decision makers' preferences in pairwise comparisons, we need to cope with the uncertainty occurring in the processes of comparisons. Such need is also, perhaps, the single most prevalent and troublesome source of difficult in all decision making processes. One of the major tasks of decision analysis is to discover ways to reduce the uncertainties to a reasonable level so that the choice or the decision can be readily made by the decision makers.

One way of reducing uncertainty is to suppress it by simplifying reality with assumed certainty. The great majority of decision making methods fall in this group. When a decision maker simplifies the decision with the assumption of certainty, he does not necessarily make any claims about his knowledge and capabilities, or about the uncertainty neglected. As long as the assumption that we have complete information and that the decision being made with assumed certainty is reasonable and acceptable, people always tend to follow this approach. The Analytic Hierarchy Process (the AHP) [Saaty, 1977, 1986, 1988] falls in this category. It assumes certainty in pairwise comparison. In the AHP, the decision makers are required to express their judgments as pairwise comparisons from 1-9 scale [Saaty, 1980]. A single numerical value (point judgment) is used for the judgment to represent the relative importance of the two objects being compared.

Another technique for coping with uncertainty is to express it explicitly by means of probabilities. We encode the decision makers' knowledge, attitudes or opinions about the likelihood of the events. Thus, uncertainties are made explicit in terms of "relative frequencies" or the decision maker's "degree of belief". This is the approach used by utility theory.

Another way of encoding uncertainty is using interval judgments [Saaty and Vargas, 1987]. This approach involves using a range, or interval of values, to represent the uncertainty in the information used by the decision maker. All the values in the range are equally acceptable, and the lower and upper bounds of the interval indicates the minimum and maximum acceptable point estimation, respectively. Interval judgments is a natural and direct way of eliciting uncertainty. In addition to allowing ambiguous judgment statements, interval judgments also allow the skipping of any unwilling judgments, in which the interval has the maximum permissible length.

3. The basic structure of interval judgments

When comparing n objects in pairwise manner with respect to a given criterion, a total of $n \times (n-1)/2$ paired comparisons is usually required. The results of the comparisons can be arranged in an $n \times n$ matrix, whose entry in the position (i,j) indicates the judgment estimation for the comparison of the object i with the object j. This matrix is called a *judgment matrix*. In the AHP, the judgment matrix for point judgments is a positive, reciprocal matrix given by

$$A = (a_{ij})_{n \times n},$$

where a_{ij} is a single numerical value and satisfies $a_{ij} \times a_{ji} = 1$ (reciprocal property) and $a_{ij} > 0$ (positive property) for all i and j. When $a_{ik} \times a_{kj} = a_{ij}$ for all i, j and k, the judgment matrix is said to be consistent. The ratio scale is derived from the matrix A by solving the eigenvalue problem $Aw = \lambda w$ for the principal right eigenvector [Saaty, 1980].

In the case of interval judgments, the entries of the judgment matrix are intervals:

$$\begin{bmatrix} 1 & [L_{12}, U_{12}] & \cdots & [L_{1n}, U_{1n}] \\ [L_{21}, U_{21}] & 1 & \cdots & [L_{2n}, U_{2n}] \\ \cdots & \cdots & \cdots & \cdots \\ [L_{n1}, U_{n1}] & [L_{n2}, U_{n2}] & \cdots & 1 \end{bmatrix}$$

where L_{ij} and U_{ij} are the lower and upper bounds of the interval judgments, respectively. Corresponding to the reciprocal property for point judgment matrix, for any $x \in [L_{ij}, U_{ij}]$, there exists $y \in [L_{ji}, U_{ji}]$ such that $x \times y = 1$, where $U_{ji} = L_{ij}^{-1}$ and $L_{ji} = U_{ij}^{-1}$, for all i and j.

Given the interval judgment matrix I, the object is to provide some idea as to the alternative that should be selected. In general, we must find the vector $w = (w_1, \ldots, w_n)$ such that

$$L_{ij} \leq f(w_i / w_j) \leq U_{ij}$$

where $f(.)$ is some function of w_i/w_j. Saaty and Vargas [1987] proposed a simulation approach. Arbel [1989] proposed a mathematical programming (LP) approach. Salo and Hämäläinën [1990] extended Arbel's LP approach, and Arbel and Vargas [1990] proposed a general non-linear programming model to compute the lower and upper bounds of each components of w. In this paper we put forth an artificial neural network model to estimate w, and compare the result from this model with the result from Arbel's and Saaty and Vargas' models.

4. The background of neural networks

A neural network consists of a number of primitive processing units linked together via weighted, directed connections. Each unit receives input signals from all its incoming units via weighted incoming connections, and it responds by signaling to all of the units to which it has outgoing connections. (See Figure 1.)[McClelland and Rumelhart, 1986] There are three functions or rules in each unit of this system: (1) the net input function, NET_i, represents the input signals to the unit i, and it is usually a function of the weighted incoming messages from all its incoming units; (2) the activation function, A_i, represents the current "state" or "activation level" of the unit, and it is determined by the inputs it received; and (3) the activation of the unit that determines its output activity, O_i.

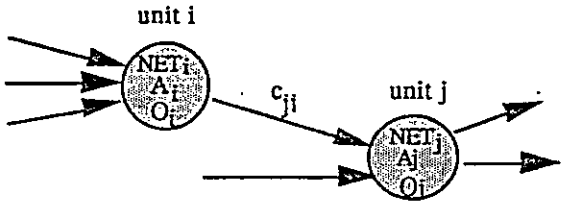

Figure 1. Processing units and connections of the neural networks

The particular neural networks we use here is called a *layered feedforward network* (See Figure 2.). These networks have three special features: (1) The input units are always the bottom layer, and the output units are the top layer. The inputs in the input layer are external input to the network, and the outputs in the top layer actually represent the output of the network; (2) The flow passes through the network only in feedforward fashion; and (3) No connections exist within a layer, i.e., there are no recursions allowed in the networks.

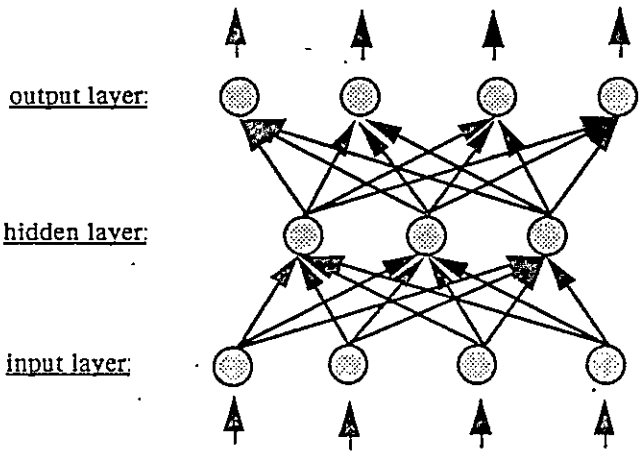

Figure 2. Feedforward layered network

Let k be the number of input units and m the number of output units of a network. Given the set

of external inputs: $I \subseteq R^k$ (k-tuple vector space) and the set of output units $O \subseteq R^m$, A network $\wp = (N, I, O, C)$ can be defined by disjoint sets N and I, where I is the input units, and a subset $O \subseteq N$ of output units. The connections are given by the set $C \subseteq (N \cup I) \times N$ of connections. With each network $\wp$, there is a function $\overline{C}$ defined on the set of connections, $\overline{C}$: $C \rightarrow R$. The function $\overline{C}$ indicates the states of the network $\wp$. For a connection $(i,j) \in E$, we denote c_{ji} as the connection strength from the unit i to unit j. We can use a matrix $\widehat{C}$ to represent connection strength values with rows and columns indexed by the set N of units. The entry in the position (i, j) of the matrix is given by c_{ji} if $(i,j) \in C$ and 0 otherwise. The feedforward condition of the network implies that $\widehat{C}$ is a lower triangular matrix with zeros in the main diagonal. Given the feedforward condition the network $\wp$ determines a function or a mapping: $FNN_{\wp} : I \rightarrow O$.

One popular way of finding such a mapping is the Backpropagation algorithm, which determines a point C_0 in the connection strength value space such that the determined network can produce the desired output.[McClelland and Rumelhart, 1986] If we use the Backpropagation algorithm, we have desired outputs: $T \subseteq R^m$ corresponding to inputs $I \subseteq R^k$. The objective is to determine a network $\wp$ whose mapping $FNN_{\wp} : I \rightarrow O$ approximates the mapping $U: I \rightarrow T$ which converts the real inputs to the real outputs. With the Backpropagation algorithm, this is achieved by the process of adjusting the state (connection strength) of the network until expected accuracy is reached. Such process is called "learning" or "training" of the network.

The Backpropagation algorithm requires a set of training patterns (examples we learn from), which is the a subset of the Cartesian product, i.e., $P_I \times P_T \subseteq I \times T$. Each pattern is a compounded point $(i_p, t_p) \in P_I \times P_T$, where i_p is the input vector and $t_p \in P_T$ is the desired output vector. Presentation of a pattern to the system corresponds to activating a set of input units.These units pass their outputs along their connections either directly to the output unit or to the intermediate (hidden) units, that relay them

onward eventually terminating in output units. The activation·pattern over the layer of output units corresponds to some particular response of the system to that input. After receiving feedback regarding the desired output pattern for each input pattern the system adjusts the connections strengths to have that input produce an output closer to the one desired. By repeatedly cycling through a set of desired input-output pairings, the system "learns" just those connections strengths that will achieve the closest match (of which it is capable) to the input-output pairings. The measure of the accuracy of the network (error function) employed by the Backpropagation algorithm is the sum of the squares of the differences between the actual and desired activations at the output units given by

$$E = \Sigma_{i,p} (t_{pi} - O_{pi})^2.$$

By minimizing the error function through a gradient descent direction of c_{ji}, we can expect to come arbitrarily close to U (if such connection strength values exist). The connection strength obtained in such manner is termed a *least mean squares* (LMS) solution. Such "error correcting" learning rule for adjusting the connection strengths has been called the LMS rule, or the delta rule.

5. A neural network model for interval judgments

With the neural network described above, we can model the process of interval judgments as a three layered feedforward network. The input layer consists of the bounds of the interval judgments L_{ij} and U_{ij}. The hidden layer represents the pairwise comparisons, and the number of hidden units is the total number of pairwise comparisons required. The output layer represents the m objects and the output of each output unit is the priority of the object that the unit represents relative to the others. The neural network for comparing four objects is given in Figure 3. The basic assumptions of this network design are as follows:

* The inputs units are linear:

$$A_j = NET_j \text{ and } O_j = A_j$$

for all the units in the input layer.

* For any unit k in the hidden and output layer, we have:

$$NET_k = \Sigma_i c_{ki} O_i ;$$

$$A_k = 1 / (1 + e^{- NET_k + \theta_k}),$$

where θ_k is the threshold of unit k. The activation function of this type is called the logistic function and has the properties of monotonicity and differentiability.

* The connections in the network can be positive, negative or zero. For ease of implementation, we assume that the network has full connections and treat the connection $(i,j) \notin C$ as nonmodifiable with zero connection strength.

The network thus defined determines a mapping $FNN_{\wp} : I \rightarrow O$, where $I \subseteq R^{24}$ and $O \subseteq R^4$. We expect to train the network such that it can produce the corresponding priority vector $w = (w_1, w_2, ..., w_n)$ for each interval judgment matrix input. This implies that we expect to approximate the mapping $U: I \rightarrow W$ by the artificially trained mapping $FNN_{\wp} : I \rightarrow O$.

Because of the particular purpose of using neural networks, we redefine the error function as follows:

$$E = \frac{1}{2} \sum_{p} \sum_{i \neq j} \left(t_{ij} - \frac{O_i}{O_j} \right)^2$$

where $t_{ij} = (L_{ij} \times U_{ij})^{1/2}$, is the "target ", "teacher", or the desired pattern for O_i / O_j. This revision means that instead of considering error as the distance between the teachers and the actual outputs in the output layer, we measured the difference between the teachers and the functions of the actual outputs in the output layer. It is clear that the result of minimizing this error function will actually force the ratio of O_i / O_j as close to the corresponding teacher t_{ij} as desired. The basic idea here is to use the geometric mean value of each judgment interval as the teachers to train the network, to find a group of connection strength values for which the ratio of outputs from each pair of output units fall within the corresponding judgment interval given by the external inputs, i.e., $L_{ij} \leq O_i / O_j \leq U_{ij}$.

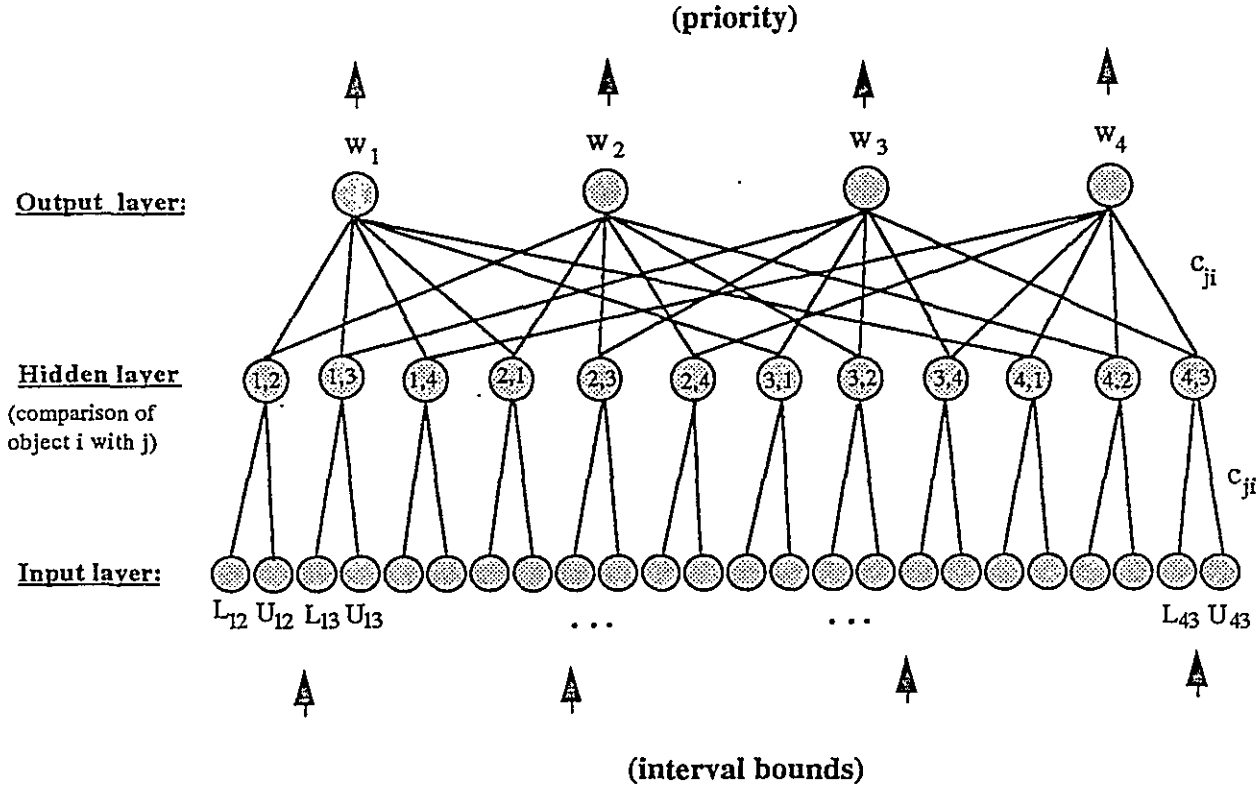

Figure 3. A neural network model for 4x4 interval judgments

With the same idea of the Backpropagation algorithm, the connections strengths are changed using the gradient:

$$\Delta_p c_{ji} = \eta \times \frac{\partial E_p}{\partial c_{ji}} = \eta \times \frac{\partial E_p}{\partial NET_{pj}} \frac{\partial NET_{pj}}{\partial c_{ji}} = \eta\ \delta_j^p\ O_{pj}$$

where $\delta_j^p = -\partial E_p / \partial NET_{pj}$ which is different from the traditional backpropagation algorithm, and η is the leaning rate denoting the adjusting scale of the connections.

There are two ways of using this revised backpropagation algorithm to derive the priority for the interval judgments:

Method 1: Using the Backpropagation for priority derivation of a single interval judgment matrix

In this method, we assume that training set contains only one pattern. With the same process of training the network, we repeatedly present the same pattern to the network and adjust the connection strengths to reduce the error. Here our only concern is the production of the desired priority for this particular interval judgment matrix and the final state of the trained network is not important. The training of the connection strengths is just a mechanism used to find the desired output for the given interval judgment matrix.

Method 2: Using the Backpropagation to train the network for priority derivation of a group of judgment matrices

This method requires reasonable sets of patterns to train the network as in the normal backpropagation algorithm applications. Our objective is to find a network state (connections strengths) to approximate the mapping $U: I \rightarrow W$. If this network state is reached, we can expect to use this network to generate the desired priority for any interval judgment inputs in the group. It is clear that we can not expect a single network to approximate all the possible judgment matrices. Our experiments also show the necessity of grouping the inputs appropriately to obtain the desired accuracy. We provide the results of our

simulation for a group of consistent interval judgment matrices in the next section. More careful investigation and intensive experimentation is necessary.

6. Simulation results

A number of simulations were performed to verify the validation and the generality of the methods proposed. All our experiments are performed on the UNIX system mainframe with the revised PDP software, written in C, which is originally provided by the PDP research group [McClelland and Rumelhart, 1988]. For both methods, we experimented with two different ways of "teacher" specifications, the arithmetic mean and the geometric mean of the bounds of the judgment intervals. The arithmetic mean reaches the solutions a little faster than the geometric mean. However, the geometric mean provides better interpretation of the results because of its relationship to the eigenvectors. We also investigated three types of error functions: Least Mean Square (LMS), Logarithm Least Mean Square (LLMS) and Hilbert's metric. The LMS is the most common error expression in most applications of the Backpropagation algorithm. However, from our experience, it is quite easy to get trapped in local minimima and it is hard to move to global optima during the process of training. In addition, because the algorithm allows the errors to grow rather than decrease in at least one dimension, it seems to-move in the wrong "direction" in the connection space instead of in the direction of the total error reduction. With the proper learning parameters and initializations, we succeeded in the cases we investigated. However, such successes are hard to predict in general. The LLMS converges to the correct solutions easier. There is, of course, no guarantee avoiding the local minima because the LLMS leads basically still a nonlinear function. Our experiments show that the LLMS method is obviously better than the LMS method for both the speed of convergence and the quality of the solutions. However, It seems to us more appropriate, theoretically, to use Hilbert's metric [Kohlberg & Pratt, 1982] as the error expression because it is the metric of the eigenvector. The implementation of this idea is currently in the stage of algorithm design and coding, and it will be part of our follow up research. For the data format, we used both point judgment data, since they are special cases of the interval judgments and easier to check the correctness of the

solutions, and interval judgment inputs. We also examined the methods for: consistent data only, inconsistent data only and combinations of both.

To illustrate the algorithms, we use for Method 1 the same example used in Arbel and Vargas's [1990]:

$$\begin{bmatrix} 1 & [2,5] & [2,4] & [1,3] \\ & 1 & [1,3] & [1,2] \\ & & 1 & [1/2,1] \\ & & & 1 \end{bmatrix}$$

The solution by the Linear Programming approach is given by:

$$w^T = (0.4679, 0.2056, 0.1470, 0.1794).$$

The results from the simulation method proposed by Saaty and Vargas [1987] are given by:

minimum	*average*	*maximum*
0.3697	*0.4702*	*0.5517*
0.1501	*0.2183*	*0.2895*
0.0929	*0.1318*	*0.1891*
0.1332	*0.1842*	*0.2600*

The result from the implementation of our proposed Method 1, with the arithmetic mean target and the LMS method after *90* presentations of the data is given by:

$$w^T = (0.4355, 0.2185, 0.1523, 0.1936).$$

This result is close to Arbel and Vargas' and satisfies the property that the ratios of w_i / w_j fall into the corresponding intervals judgments with only one deviation of less than *0.002*. With the geometric mean of the teacher and the LLMS method, the results are given by:

$$w^T = (0.4501, 0.2128, 0.1396, 0.1975),$$

without any deviation.

For the logarithm error function and the geometric mean, we use point judgment matrices to estimate the approximation of the solutions to the corresponding principal eigenvectors. For the following consistent matrix:

$$\begin{bmatrix} 1 & 3 & 6 & 3 \\ 1/3 & 1 & 2 & 1 \\ 1/6 & 1/2 & 1 & 1/2 \\ 1/3 & 1 & 2 & 1 \end{bmatrix}$$

we obtained the solution $w^T = (0.5461\ 0.1825\ 0.0904\ 0.1810)$ with a total squared error equal to 0.0028 after 50 presentations of the matrix.

For Method 2, we trained the network with a pattern set (of size 10) of consistent interval judgment matrices with the logarithm mean square function and geometric mean targets. A total error of 0.08857 was obtained after 450 of the presentations of each interval judgment matrix data. The connection strengths are given by:

		input layer	
		lower bounds	upper bounds
	(1)	-1.230123	-2.675019
	(2)	-1.734057	-2.944314
	(3)	-3.581413	-11.58413
	(4)	-3.011533	-1.667727
	(5)	-1.311551	-4.631078
hidden layer	(6)	-1.315490	-2.623856
	(7)	-0.515770	-1.588770
	(8)	-0.980903	-3.141418
	(9)	-0.610185	-1.945769
	(10)	-1.620738	-3.990551
	(11)	-1.904183	-3.444780
	(12)	0.559310	-1.464178

		output layer				
		(1)	(2)	(3)	(4)	thresholds
hidden layer	(1)	1.689003	5.064165	0.000000	0.000000	-1.009856
	(2)	2.672118	0.000000	2.473513	0.000000	-0.615173
	(3)	3.384423	0.000000	0.000000	2.938550	-4.021264
	(4)	1.389909	-0.376752	0.000000	0.000000	-5.774281
	(5)	0.000000	2.130989	-0.435712	0.000000	-1.464055
	(6)	0.000000	2.114039	0.000000	0.822433	-1.197417
	(7)	2.114224	0.000000	-1.874608	0.000000	-1.527127
	(8)	0.000000	0.467882	-0.031106	0.000000	-2.906212
	(9)	0.000000	0.000000	-0.006308	0.369275	-2.190239
	(10)	1.736171	0.000000	0.000000	-0.440586	-3.517609
	(11)	0.000000	2.583185	0.000000	0.264742	-3.291111
	(12)	0.000090	0.000000	2.087249	-1.000665	0.430566
	thresholds	8.442033	-0.239755	-0.956069	-0.174941	

We now use these connections to generate the priority vector for two matrices. One of them, used by Arbel and Vargas [1990] belonged to the training set. The priority vector is given by

$$w^T = (0.4592\ 0.2056\ 0.1388\ 0.1964).$$

The second matrix given by:

$$\begin{bmatrix} 1 & [1,4] & [5,6] & [1,4] \\ [1/4,1] & 1 & [1,2] & [1/2,2] \\ [1/6,1/5] & [1/2,1] & 1 & [1/3,1] \\ [1/4,1] & [1/2,2] & [1,3] & 1 \end{bmatrix}$$

which does not belong to the training set, and the priority vector is given by

$$w^T = (0.4736\ 0.2008\ 0.1144\ 0.2112).$$

7. CONCLUSION

We have described a new method to derive priorities for interval judgments matrices. With the proposed method, the problem can be modeled and analyzed with a simple mechanism and handled with relative ease. Once a neural network model is constructed and implemented by the training process, it is easy to experiment with the network to ascertain its behavior under different assumptions and a variety of input patterns. Using the simulation of the neural network, it is possible to obtain approximate solutions to the problems, especially in the inconsistent case, in which the other available methods become infeasible or difficult from a practical standpoint. A shortcomings of the neural network approach is that the solutions obtained are only approximation. In addition, it could be expensive, in the sense of computational time, because of problems we encountered with local minima.

REFERENCES

Arbel, A., (1989) *"Approximate articulation of preference and priority derivation,"* Eur . J. Oper. Res., No.43, pp.317-326.

Arbel, A., & Vargas, L. G., (1990) *"The Analytic Hierarchy Process with Interval judgments,"* In Proceedings of the MCDM conference held in Washington, D. C., August 1990

Kohlberg, E., & Pratt, E. W., (1982) *"The Contraction Mapping Approach to the Perron-Frobenius Theory: Why Hilbert's Metric?"*, Math. of Oper. Res., Vol.7., No.2, May 1982, pp.198-210.

McClelland, J. L., & Rumelhart, D., (1986) *Parallel Distributed Processing (Vol. 1 and 2)*. Cambridge, MA: MIT Press.

McClelland, J. L., & Rumelhart, D., (1988). *Exploration in Parallel Distribution Processing: A Handbook of Models, Programs and Exercises*, The MIT Press, Cambridge, MA.

Saaty, T. L.(1980) *The Analytic Hiereachy Process.* McGraw-Hill, New York.

Saaty, T. L., & Vargas, L. G., (1987) *"Uncertainty and Rank order in the Analytic Hierarchy Process"*, Eur. J. Oper. Res., No.2, pp.107-117.

Salo, A. and Hämäläinën, R. P., (1990). *"Decision support under ambiguous preference statements"*, Proc. of the Annual Conference of the Operational Research Society of Italy, Models and Methods for decision Support, October 1990, pp.229-243.