TECHNICAL ARTICLE

SIMULATION 74:3, 128-137

ISSN 0037-5497/00
Printed in the United States of America

Input Data Analysis Using Neural Networks

Anil Yilmaz
Turkish Prime Ministry
State Planning Organisation
Yucetepe, Ankara 06100, Turkey
E-mail: anil@dpt.gov.tr

Ihsan Sabuncuoglu
Department of Industrial Engineering
Bilkent University
Bilkent, Ankara 06533, Turkey
E-mail: sabun@bilkent.edu.tr

Simulation deals with real-life phenomena by constructing representative models of a system being questioned. Input data provide a driving force for such models. The requirement for identifying the underlying distributions of data sets is encountered in many fields and simulation applications (e.g., manufacturing economics, etc.). Most of the time, after the collection of the raw data, the true statistical distribution is sought by the aid of nonparametric statistical methods. In this paper, we investigate the feasibility of using neural networks in selecting appropriate probability distributions. The performance of the proposed approach is measured with a number of test problems.

Keywords: Input data analysis, neural networks, probability distribution functions

1. Introduction

Simulation models have a very wide range of application areas from manufacturing to defense, economic and financial systems, and the input data used in these models are usually represented by probability distribution functions. Since input data provides a driving force for simulation models, this topic is extensively studied in the simulation literature [1]. As also indicated by Law and Kelton [2], failure to choose the correct distribution can affect credibility of simulation models. However, identification of the true underlying statistical distribution for a given data set is a difficult task for a simulation analyst.

In general, there are four steps in the input data analysis [3]:

1. Collection of the raw data,
2. Identification of the candidate distributions,
3. Estimation of the parameters of these distributions,
4. Testing the distribution assumptions and the related parameters by goodness-of-fit tests.

This study mostly aims at the second step of the above procedure. At this stage, after collecting the raw data, practitioners seek an underlying statistical

distribution by the aid of nonparametric statistical methods (heuristics and other graphical methods). Summary statistics such as minimum, maximum, mean, median, variance, coefficient of variation, lexis ratio, skewness, kurtosis, etc., are used, as well as other statistical tools, some of which are histograms, line graphs, quantile summaries, box plots, Q-Q and P-P plots. In practice, this task is sometimes cumbersome and time consuming.

The aim of this study is to investigate the feasibility of using neural networks for the input data analysis (identification of probability distributions) and discuss the difficulties in using neural networks, as well as their strength and weaknesses over the traditional methods (i.e., Chi-square goodness-of-fit test, etc.).

The rest of the paper is organized as follows. In Section 2, we present a brief review of the relevant literature on the application of neural networks to the input data analysis. In Section 3, we explain the methodology used in our study. We give the experimental settings in Section 4. The computational results are discussed in Section 5. Finally, we make concluding remarks and suggest further research directions.

2. Literature Survey

The input data analysis, which is also referred to as input data modelling or modelling input processes, is not extensively studied in the simulation literature. The topic is discussed in detail in [1], [2] and [3]. General procedures to identify the correct distribution functions are also outlined in these references. In the input data analysis literature, Shanker and Kelton [5] investigated the effect of distribution selection on the validity of output from single queuing models. The authors also compared the empirical distribution functions (i.e., distribution of the sample data) with standard parametric distribution functions (e.g., Uniform, Exponential, Weibull). Their results indicated that on the basis of variance and bias in their estimations, the performance of the empirical distributions is comparable with, even sometimes better than, standard distribution functions. Vincent and Law [6] proposed a software package called UNIFIT II for input data analysis. The authors discuss the role of simulation input modeling in a successful simulation study. In a related work, Vincent and Kelton [7] investigated the importance of input data selection on validity of simulation models and discuss the philosophical aspects of the current thinking. Johnson and Mollaghasemi [8] explored the topic from a statistical point of view. The authors provided a comprehensive bibliography and a list of specific research problems in the input data analysis. Finally, Banks, Gibson, Mauer and Keller [9] discussed empirical versus thoretical distributions and expressed their opposite views (points and counterpoints) on input data analysis.

In the neural network literature, neural networks can be used in place of statistical approaches applied to classification and prediction problems [10]. In general, advantages of neural networks in statistical applications are their ability to classify robustness to probability distribution assumptions, and the ability to give reliable results even with incomplete data. In this context, neural networks are employed where regression, discriminant analysis, logistic regression or forecasting approaches are used.

Marquez [11] has provided a complete comparison of neural networks and regression analysis. The results of his study suggest that the neural networks can do fairly well in comparison to regression analysis. The prediction capability of neural networks has been studied by a large number of researchers. In early papers, Lapeds and Farber [12] and Sutton [13] offered evidence that the neural models are able to predict time series data fairly well. Many comparisons of neural networks and time series forecasting techniques, such as the Box-Jenkins approach, are reported [10]. The reader can refer to [14], [15] and [16] for further reading on application of neural networks to data analysis.

In the literature, there are only a few studies on the application of neural networks to the input data analysis problem (Table 1). The first study in this area is by Sabuncuoglu, Yilmaz and Oskaylar [17], who investigated the potential applications of neural networks

Table 1. A list of previous studies and their characteristics

Publications	Type of Neural Network	Distributions Considered	Input Data Representation
Sabuncuoglu, Yilmaz and Oskaylar (1992)	Back Propagation and Counter Propagation	Exponential, Uniform, Normal	Histograms
Akbay, Ruchti and Carlson (1992)	Back Propagation and Probabilistic	Uniform, Exponential, Weibull, Gamma, Lognormal, Normal, Beta	Quantiles
Aydin and Ozkan (1996)	Multi Layer Perceptron	Normal, Gamma, Exponential, Beta	Minimum, Maximum, Normalized Frequencies
Yilmaz and Sabuncuoglu (1997)	Probabilistic	Uniform, Exponential, Weibull, Gamma, Lognormal, Normal, Beta	Skewness, Quantiles, Cumulative Probabilities

during the input data analysis stage of simulation studies. Specifically, counter-propagation and back-propagation networks were used as the pattern classifier to distinguish data sets among three basic distribution functions: exponential, uniform and normal. Histograms consisting of ten equal-width intervals were used as input vectors in the training set. The performance of the networks was compared to the standard goodness-of-fit tests for different sample sizes and parameters. The results indicated that neural networks are quite successful for identification of these three distribution functions.

Akbay, Ruchti and Carlson [18] proposed a neural network model, which is based on the quantile information to recognize certain patterns in raw data sets. The authors measured the prediction capability of a probabilistic and a back-propagation neural network and compared the results with traditional statistical methods. Nine equal interval normalized quantile values were used as the input, and 25 different categories of distributions were identified. The results indicated that the probabilistic neural network (PNN) learned (i.e., was able to correctly identify) all the 25 categories in the training set, whereas the back-propagation network was able to learn 24 categories.

In another study, Aydin and Özkan [19], using a multi-layer perceptron network, investigated the performance of the neural network for distinguishing among normal, gamma, exponential and beta distributions. They compared the results with those of the chi-square test. The input used for training the networks was selected as the minimum and maximum values for the distributions, as well as normalized frequencies. The number of frequency intervals to be used was determined by constructing various networks with different numbers of frequency intervals.

In a recent study, Yilmaz and Sabuncuoglu [20] developed a PNN to distinguish 23 different types of seven probability distributions. The authors used skewness, eight quantile and twelve cumulative probability values to train the neural network. Their results showed that PNN is good at hypothesizing the distribution of raw data sets. The authors also suggested that there should be a grouping of distributions with similar shapes and a specialized neural network should implement the selection process within each group.

Since in this study multiple neural networks are used for input modelling, it is necessary to review the related work in modular neural networks. In what follows we review these applications.

Hashem and Schmeiser [21] used multiple neural networks with different architectures to achieve acceptable model accuracy. Specifically, authors represented a single neural network by a linear combination of several trained networks to minimize the mean square error. Rogova [22] employed multiple neural networks for optical character recognition. Results indicated that the combined neural network leads to a 15% to 30% reduction in the error compared to the best individual classifier. In another study, Jordan and Jacobs [23] proposed an architecture, which is a hierarchical mixture model of experts and expectation maximization algorithms. By this approach, the authors divided a complex problem into simpler problems that can be solved by separate expert networks. Boers and Kuiper [24] developed a computer program to find a modular artificial neural network for a number of application areas (handwritten digit recognition, mapping problem, etc.). In a later work, Hashem [25] extended the idea of optimal linear combinations of neural networks and derive closed form expressions. The results demonstrated considerable improvements in model accuracy, leading to a 81% to 94% reduction in true MSE compared to the apparent best neural network. The author also provided a comprehensive bibliography on multiple neural networks.

Yang and Chang [26] proposed a two-phase learning modular neural network architecture to transform a multimodal distribution into known and more learnable distributions. They decomposed the input space into several subspaces and trained a separate multilayer perceptron for each group. A global classifier network is trained for the second phase of learning. This network uses the inputs from various local networks and maps this new data set to a final classification space. The authors concluded that the two-phase learning modular network architecture reduces to a great extent the chance of sticking to a local minimum. They also argue that the two-phase method is better in performance and more robust, and less dependent on architecture parameters as well as selection of training samples.

Chen et al. [27] presented a self-generating modular neural network architecture to implement the divide-and-conquer principle. A tree-structured modular neural network is automatically generated by recursively partitioning the input space. The results on several problems, compared to a single multi-layer perceptron, indicated that the proposed method performs well both in terms of high success rate and short CPU time.

3. Research Methodology

We aim at differentiating among 23 different special types of seven distinct distributions based on different shape parameters. These distributions were selected because they are frequently used in simulation studies.

Based on our previous experience, after a few months of experimentation with various types of neural networks, different sample sizes and many input combinations, we achieved a moderate success (as discussed in [20]). In the above experiments, all the 23 distributions were tackled at the same time. In other words, a single neural network was used to choose an appropriate distribution function for a given data set. Since only a moderate success was achieved in the previous study, the authors recommended searching

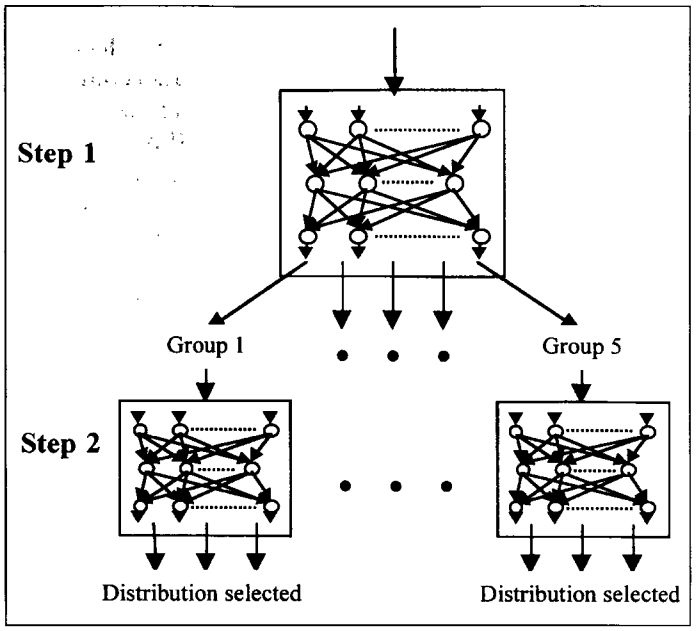


Figure 1. Two-step multiple neural network approach

for multiple neural networks.

According to this approach, in the first step a single network is used to classify distributions with similar shapes. In the second step, specialized networks are used to detect different types from each group of distribution functions. In this paper we implement this two-step multiple neural network approach (Figure 1). Step 1 consists of grouping distributions that have similar shapes and training a neural network that performs the classification task based on this grouping. Here, the trained neural network is expected to correctly categorize among the different groups of distributions.

In the second step, for each group of distributions identified in the previous step, a different network is trained and tested. These specialized networks are used to further classify the input data into specific distribution functions. At this stage, the training sets and network structures for each group are formed by trial and error. The inputs that would be most appropriate for each group are selected among all possible summary statistics such as range, mean, variance, coefficient of variation, skewness, kurtosis, quantile and cumulative probability information. In addition, composite measures such as kurtosis divided by the coefficient of variation or (skewness + kurtosis)/(coefficient of variation) are used in the experiments. This selection process is explained in detail in the following section.

After training the neural networks, their performances are measured by randomly generated data of known distributions.

4. Experimental Setting

4.1 Distributions Considered

There are seven distinct distributions used in this study: Uniform, Exponential, Weibull, Gamma, Lognormal, Normal and Beta. These distributions are selected because they are frequently encountered in scientific literature and real-life applications. Based on the different shape parameters, we use three types of Weibull, Gamma and Lognormal distributions. Similarly, eleven different types of Beta distribution are considered corresponding to different shape parameters. Totally, 23 distributions are used in the experiments (Table 2).

4.2. Neural Network Types and Structures

We initially consider three neural network types: back-propagation, counter-propagation and probabilistic neural networks [28]. Based on extensive computational experiments, however, we eliminated the back-propagation and probabilistic neural networks due to their inferior performance. Hence, we mainly focus on the counter-propagation network.

A counter-propagation network constructs a mapping from a set of input vectors to a set of output vectors acting as a hetero-associative nearest-neighbour classifier [29]. Its applications include pattern classification, function approximation, statistical analysis and data compression.

When presented with a pattern, the trained counter-propagation network classifies that pattern into a

Table 2. Distributions used in this study

	Distribution /Type	Parameters
1	Uniform	min=0, max=1
2	Exponential	location=0, shape=1
3	Weibull-1	location=0, scale=1, shape=0.5
4	Weibull-2	location=0, scale=1, shape=2
5	Weibull-3	location=0, scale=1, shape=3
6	Gamma-1	location=0, scale=1, shape=0.5
7	Gamma-2	location=0, scale=1, shape=2
8	Gamma-3	location=0, scale=1, shape=3
9	Lognormal-1	location=0, scale=0, shape=0.5
10	Lognormal-2	location=0, scale=0, shape=1
11	Lognormal-3	location=0, scale=0, shape=1.5
12	Normal	mean=0, variance=1
13	Beta-1	min=0, max=1, shape1=1.5, shape2=5
14	Beta-2	min=0, max=1, shape1=1.5, shape2=3
15	Beta-3	min=0, max=1, shape1=5, shape2=1.5
16	Beta-4	min=0, max=1, shape1=3, shape2=1.5
17	Beta-5	min=0, max=1, shape1=3, shape2=3
18	Beta-6	min=0, max=1, shape1=5, shape2=5
19	Beta-7	min=0, max=1, shape1=2, shape2=2
20	Beta-8	min=0, max=1, shape1=0.8, shape2=2
21	Beta-9	min=0, max=1, shape1=1, shape2=2
22	Beta-10	min=0, max=1, shape1=2, shape2=0.8
23	Beta-11	min=0, max=1, shape1=2, shape2=1

particular group by using a stored reference vector; the target pattern associated with the reference vector is then output. The input layer acts as a buffer. The network operation requires that all the input vectors have the same length, and hence input vectors are normalized to one. As discussed in [30], counterpropagation combines two layers from different paradigms. The hidden layer is a Kohonen layer, with competitive units that perform unsupervised learning. The processing elements in this layer compete such that the one with the highest output is activated. The top layer is the Grossberg layer, which is fully interconnected to the hidden layer. Since the Kohonen layer produces only a single output, this layer provides a way of decoding that output into a meaningful output class. The Grossberg layer is trained by the Widrow-Hoff learning rule.

4.3 Network Construction, Training and Testing

As discussed in the previous section, the work is carried out in two consecutive steps.

Step 1 (Grouping the Distributions):

The distribution functions (given in Table 2) are grouped into six categories based on their shapes. The resulting groups are shown in Figure 2. Even though

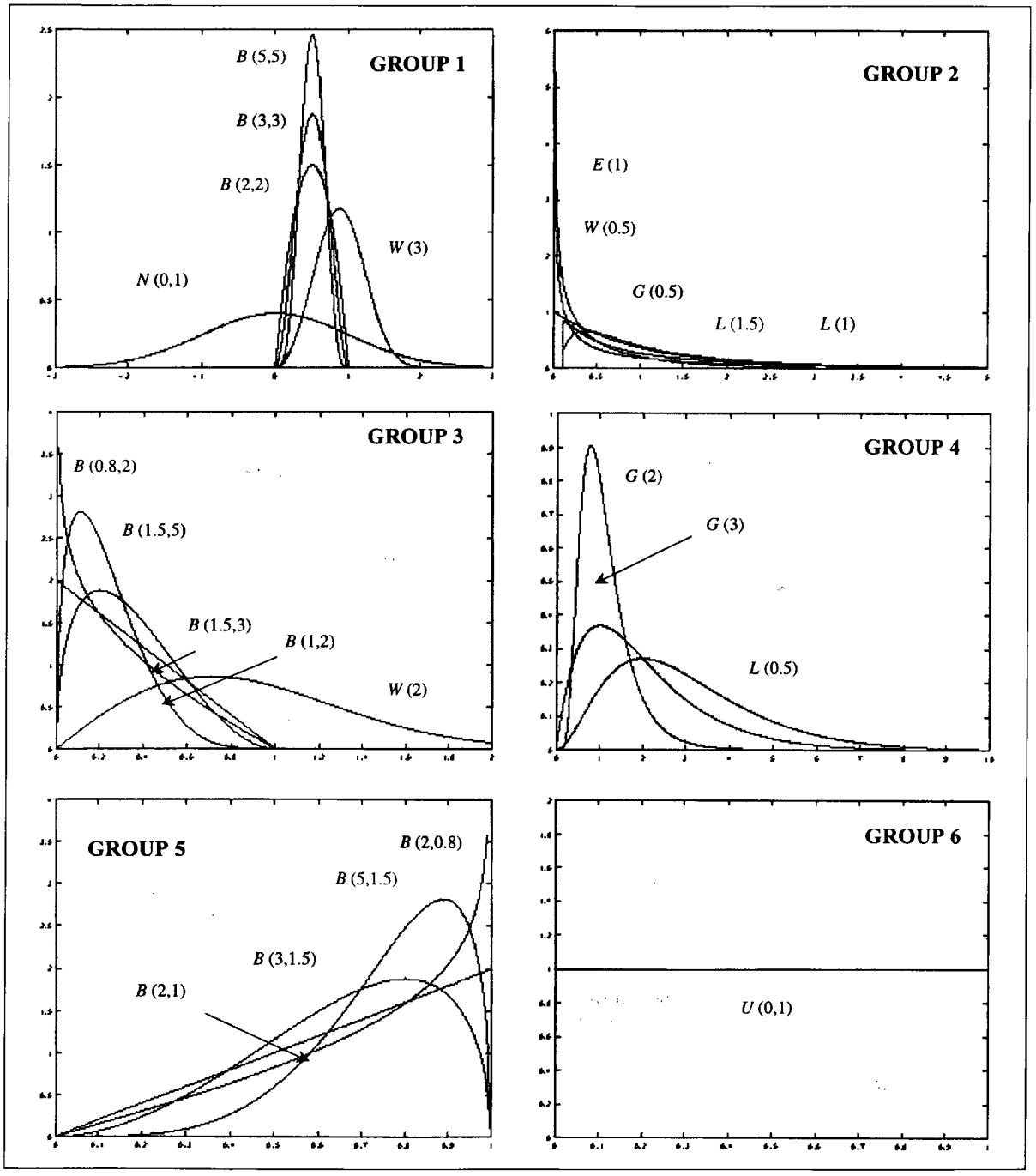


Figure 2. Groups of distribution functions

clustering techniques or unsupervised neural networks could have been used for grouping, we performed this step manually. First, we formed preliminary groups visually by considering their general shapes (e.g., Group 1 represents bell-shaped distributions, Group 2 consists of right-skewed distributions, etc.). Then we looked at skewness, kurtosis, quantiles and cumulative probabilities of these distributions and finalized the grouping. As can be seen in Appendix 1, skewness and quantiles of the different groups differ from each other, whereas the distributions in each group have very close parameters values.

After forming the above groups, the training set is prepared. For this purpose, we use the UNIFIT-2 Statistics Package [31]. All the possible theoretical summary statistics for each of the 23 distributions are investigated on the experimental basis in order to find the inputs that are useful for the network to distinguish among groups. After numerous experiments, skewness and quantile information (measured at seven different points) are found to be the best characterizing statistics. The training set is given in Appendix 1. Note that some distributions in these groups are duplicated to form equal-size groups. Hence, equal numbers of examples are presented to the network to achieve a balanced training.

The proposed counter-propagation network has eight neurons corresponding to eight inputs in the input layer. To determine the number of neurons in the hidden (or Kohonen) layer is a difficult task and is usually done by experimentation. When there are too many neurons, the network memorizes, and its ability to generalize gets weaker. On the other hand, using too few neurons causes the network not to learn. After carrying out some experiments and considering the above concerns, the number of processing units in the Kohonen layer is determined to be fifteen. There are six neurons in the output (or Grossberg) layer corresponding to six groups of distributions.

The counter-propagation network is successfully trained (i.e., the root mean square (RMS) error converged to zero) by using the training set given in Appendix 1. Specifically, it learned all the examples in the training set after 5,000 iterations. In order to test the network performance, a test set is prepared. For each of the 23 distributions, five raw data sets of sample size 100 are randomly generated. The resulting 115 data sets are processed by a Pascal program and are transformed into test examples, each represented by one skewness and seven quantile values. When the test set is presented to the trained neural network, it is observed that almost all test examples are correctly identified. The network fails for only three out of 115 examples. Hence, at this stage, we concluded that Step 1 of the proposed procedure is successfully implemented.

Step 2 (Identification of Distributions):

In the second step, we train a different neural network for five groups. (Since the sixth group is uniform itself, there is no need to train a network)

Each group has its own attributes (characteristics). Therefore, identifying different distributions within each group necessitates the use of different input representation for each network. The inputs that are most suitable for each of the five groups are identified experimentally in the same way as discussed in Step 1. The training set for each of the five groups is given in Appendix 2.

The topology of the counter-propagation network varies among groups. The number of input layer neurons is determined by the number of inputs in the training examples. Also, the number of neurons in the hidden layer for each group is found on the experimental basis. Here, we observed that it would be suitable to use twice as many neurons as the number of distributions to be identified in each group. The number of output neurons is determined by the number of distributions that form the groups.

All the five neural networks are successfully trained as the RMS errors converge to zero after 5,000 iterations. The trained networks are tested by the same data sets generated in Step 1. Again, the raw data sets are transformed into appropriate test examples by the Pascal computer program. The results of the tests are discussed in detail in the next section.

5. Results

Having trained the neural networks successfully, we measure their performances by the test data sets of sample size 50, 100 and 500 (a total of 345 test examples).

In Step 1, when the test data sets are presented to the trained counter-propagation network, it identified the correct grouping with 97.4% success (Table 3). All the examples, which belong to Groups 1, 3, 5 and the uniformly distributed set (Group 6), were perfectly categorized. For Group 2, 24 out of 25 sets were successfully classified, whereas the success rate was 13 out of 15 for Group 4.

In general, we observed that the neural network performance improves as sample size increases (see Table 3). It can also be noted that the success rate in Step 1 is higher than in Step 2. This is expected because neural networks used in Step 2 have to distinguish specific distributions among similar distributions, whereas the neural network used in Step 1 just classifies the distributions among more distinct groups.

By examining the results in Table 3, we can conclude that neural networks should not be recommended for small sample sizes; the success rate of the two-step neural network approach is around 58% for a sample size of 50, but it improves considerably to

Table 3. Test results for the neural networks

STEP 1 RESULTS				
Network	**Input Representation**	**Success Rate (%)**		
CNN used for step 1	S and 7 Q values	70.4	97.4	95.7
STEP 2 RESULTS				
Groups	**Input Representation**	**Success Rate (%)**		
1. Symmetric (Bell type)	K/CV, R	68	76	100
2. Skewed to right (sharp)	I/CV, R	40	72	80
3. Skewed to the right (regular)	R, S and 0.125 Q	48	88	96
4. Skewed to right (mild)	R, K and 0.5 Q	53	86.7	93
5. Skewed to the left	I/CV, 0.5, 0.75, 0.875 and 0.9 Qs	45	80	100
6. Symmetric-Uniform	Already identified in step 1, so no network is used for step 2	80	100	100
OVERALL RESULTS				
Groups		**Success Rate (%)**		
1. Symmetric (Bell type)		64	80	100
2. Skewed to the right (sharp)		44	72	80
3. Skewed to the right (regular)		68	88	88
4. Skewed to the right (mild)		27	86.7	80
5. Skewed to the left		70	95	100
6. Symmetric-Uniform		100	100	100
TOTAL		**58.3**	**84.3**	**90.4**

S: skewness **CV:** coefficient of variation
Q: quantile **R:** range
K: kurtosis

84% and 90% when the sample size is increased to 100 and 500, respectively.

From the results it appears that the neural networks perform fairly well to distinguish Group 6 (uniform), Group 5 (skewed to the left) and Group 3 (skewed to the right-regular). These are followed by Group 1 (symmetric), Group 4 (skewed to the right-mild) and Group 2 (skewed to the right-sharp). Note that Groups 3 and 5, for which the neural network was more successful, consist of mostly beta distributions with different shape parameters. Group 6 (uniform) is also a special type of beta distribution. This means that neural networks are quite successful in distinguishing beta distributions. To some extent, the ability of the neural networks to distinguish the beta distribution from the others also continues for Group 1 (symmetric-bell type).

It seems that the second group which includes exponential, Weibull, Lognormal and Gamma distributions, is the most difficult group for our neural network approach. Note that this group consists of very skewed distributions (skewed to the right) which apparently created a great deal of difficulty for the neural model.

Results also indicated that the two-step multiple neural network approach proposed in this paper is more successful than the one-step single neural network approach discussed in [20].

As seen in Table 4, the percentage of improvement by the two-step approach is lowest for small sample sizes, moderate for large sample sizes and highest for medium sample size (n = 100).

The multiple neural network approach proposed in this paper and traditional goodness-of-fit tests (GFT) are not directly comparable, because the proposed approach is a meta model which selects a distribution for the given data set (i.e., rejects all the other candidate distribution functions), whereas GFT is more an analysis tool which tests if a candidate distribution is a good fit for the data set. (This concept is illustrated in Figure 3 where D_i represents the i-th distribution function and S_i corresponds to the i-th step of the multiple neural network approach). While doing that, GFT might require more than one iteration for testing candidate distributions. It is also quite possible that GFT might reject the true underlying distributions. In our case, for example, a chi-square goodness-of-fit test applied to data sets rejected eleven and six distributions for sample sizes 50 and 100, respectively. This means that this technique is less reliable when the sample size is small.

Another distinguishing characteristic of the neural network approach from GFT is that once a distribution is selected, other alternative distributions are rejected. On the other hand, more than one distribution can

Table 4. Success rate for the two approaches

	Sample Size		
	50	**100**	**500**
One-step procedure	50.0%	62.6%	80.0%
Two-step procedure	58.3%	84.3%	90.4%

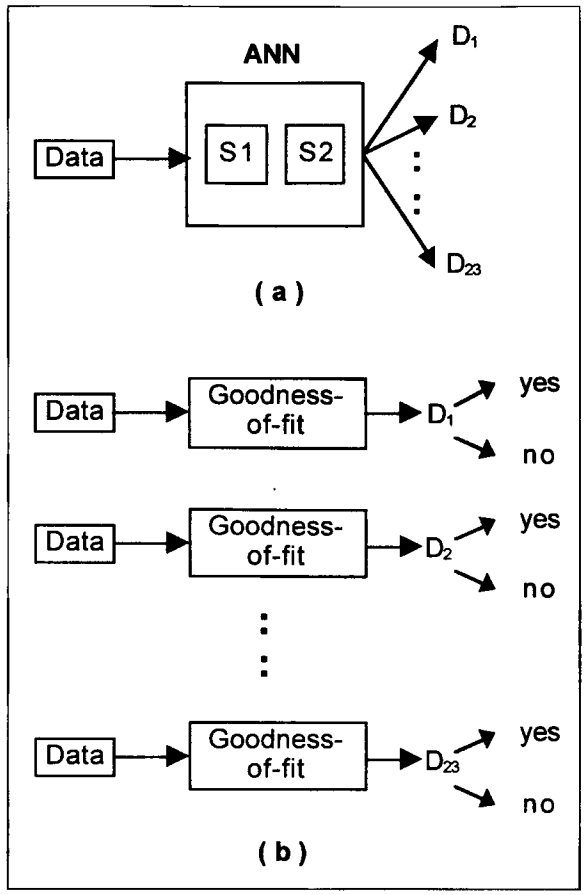


Figure 3. ANN versus goodness-of-fit tests

easily pass the test in the classical GFT approach. Hence, the results of the GFT test may not always be conclusive. In that respect, GFT and neural networks should be considered as complementary techniques. Specifically, the results of the neural network (i.e., distribution recommended by the neural network) can be used by the GFT to make more reliable and quicker decisions.

6. Concluding Remarks

In this paper, we developed a multiple neural network architecture to select probability distribution functions. The results indicated that the multiple neural network approach is more successful than the one-step single neural network approach in identifying distributions.

In this study, we also analysed the strengths and weaknesses of neural networks relative to the traditional GFT approach. Our conclusion is that the neural networks can be successfully used in simulation input data analysis as a quick reference model. In this context, neural networks can complement the function of the traditional GFT approach (i.e., the suggested reference models can be further analysed by the traditional methods).

Even though some groundwork has been established in this paper, there are several research issues that need to be addressed in future studies. First, neural networks can be trained to act as a traditional GFT (Figure 3(b)). In this case, one special neural network is trained for each distribution function and is used to accept or reject the hypothesis. Second, the performance of the neural network approach in this problem domain can be improved by using different NN architectures. Third, unsupervised neural networks can be used to form the groups. Finally, neural networks can be used in estimating the parameters of the distributions. This may be a fruitful future research area for neural networks in the field of probability distribution selection.

7. References

[1] Vincent, S.G. "Input Data Analysis." In *Handbook of Simulation*, J. Banks (ed.), pp 55-91, 1998.

[2] Law, A. and Kelton, W.D. *Simulation Modeling and Analysis*, Second Edition, McGraw-Hill, 1991.

[3] Banks, J., Carson, J.S. and Nelson, B.L. *Discrete Event System Simulation*, Second Edition, Prentice- Hall, 1996.

[4] Bratley P., Fox, B.L. and Schrage, L.E. *A Guide to Simulation*, Second Edition, Springer-Verlag, New York, 1987.

[5] Shanker A., and Kelton, W.D. "Empirical Input Distributions: An Alternative to Standard Input Distributions in Simulation Modeling." *Proceedings of the 1991 Winter Simulation Conference*, B.L. Nelson, W.D. Kelton and G.M. Clark (eds.), pp 978-985, 1991.

[6] Vincent, S.G. and Law, A.M. "Unifit II: Total Support for Simulation Input Modelling." *Proceedings of the 1992 Winter Simulation Conference*, J.J. Swain, D. Goldsman, R.C. Crain and J.R. Wilson (eds.), pp 136-142, 1991.

[7] Vincent, S.G. and Kelton, W.D. "Distribution Selection and Validation." *Proceedings of the 1992 Winter Simulation Conference*, J.J. Swain, D. Goldsman, R.C. Crain and J.R. Wilson (eds.), pp 300-304, 1992.

[8] Johnson, M.E. and Mollaghasemi, M. "Simulation Input Data Modelling." *Annals of Operations Research*, Vol. 53, pp 47-75, 1994.

[9] Banks, J., Gibson, R.R., Mauer, J. and Keller, L. "Simulation Input Data: Point-Counterpoint." *IIE Solutions*, January 1998, pp 28-36, 1998.

[10] Sharda, R. "Neural Networks for the MS/OR Analyst: An Application Bibliography." *Interfaces*, Vol. 24, pp 116-30, 1994.

[11] Marquez, L.O. "Function Approximation Using Neural Networks: A Simulation Study." PhD Dissertation, University of Hawaii, Honolulu, HI, 1992.

[12] Lapeds, A. and Farber, R. "Nonlinear Signal Prediction Using Neural Networks: Prediction and System Modeling." LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, NM, 1987.

[13] Sutton, R.S. "Learning to Predict the Method of Temporal Differences." *Machine Learning*, Vol. 3, No. 1, pp 9-44, 1988.

[14] Ali, D.L., Ali, S. and Ali, A.L. "Neural Nets for Geometric Data Base Classifications." *Proceedings of the SCS Summer Simulation Conference*, pp 886-890, 1988.

[15] Pham, D.T. and Oztemel, E. "Control Chart Pattern Recognition Using Neural Networks." *Journal of Systems Engineering*, Vol. 2, pp 256-262, 1992.

[16] Udo, G.J. and Gupta, Y.P. "Applications of Neural Networks in Manufacturing Management Systems." *Production Planning and Control*, Vol. 5, No. 3, pp 258-270, 1994.

[17] Sabuncuoglu, I., Yilmaz, A. and Oskaylar, E. "Input Data Analysis for Simulation Using Neural Networks." In *Proceedings of the Advances in Simulation '92 Symposium*, A.R. Kaylan and T.I. Ören (eds.), pp 137-150, 1992.

[18] Akbay, K.S., Ruchti, T.L. and Carlson, L.A. "Using Neural Networks for Selecting Input Probability Distributions." *Proceedings of ANNIE'92*, 1992.

[19] Aydin, M.E. and Özkan, Y. "Daϑylym Türünün Belirlenmesinde Yapay Sinir Aϑlarynyn Kullanylmasy." *Proceedings of the*

First Turkish Symposium on Intelligent Manufacturing Systems, pp 176-184, 1996.

[20] Yilmaz, A. and Sabuncuoglu, I. "Probability Distribution Selection Using Neural Networks." *Proceedings of the European Simulation Multiconference '97*, 1997.

[21] Hashem, S. and Schmeiser, B. "Improving Model Accuracy Using Optimal Linear Combinations of Trained Neural Networks." *IEEE Transactions on Neural Networks*, Vol. 6, No. 3, pp 792-794, 1995.

[22] Rogova G. "Combining the Results of Several Neural Network Classifiers." *Neural Networks*, Vol. 7, No. 5, pp 777-781, 1994.

[23] Jordan, M.I. and Jacobs, R.A. "Hierarchical Mixtures of Experts and the EM Algorithm." *Neural Computation*, Vol. 6, pp 181-214, 1994.

[24] Boers, E.J.W. and Kuiper, H. "Biological Metaphors and the Design of Modular Artificial Neural Networks." Masters Thesis, Departments of Computer Science and Experimental and Theoretical Psychology at Leiden University, The Netherlands, 1992.

[25] Hashem, S. "Optimal Linear Combinations of Neural Networks." *Neural Networks*, Vol. 10, No. 4, pp 599-614, 1997.

[26] Yang, S. and Chang, K. "Multimodal Pattern Recognition by Modular Neural Network." *Optical Engineering*, Vol. 37, No. 2, pp 650-659, 1998.

[27] Chen, K., Yang, L., Yu, X. and Chi, H. "A Self-Generating Modular Neural Network Architecture for Supervised Learning." *Neurocomputing*, Vol.16, pp 33-48, 1997.

[28] Bose, N.K. and Liang, P. *Neural Network Fundamentals with Graphs, Algorithms, and Applications*, McGraw-Hill, 1996.

[29] NeuralWare, Inc. *Neural Computing*, Pittsburgh, 1991.

[30] Hecht-Nielsen, R. *Neurocomputing*, Addison-Wesley, 1990.

[31] Law, A. and Vincent, S.G. *Unifit II User's Guide*. Averill M. Law & Associates, Tucson, AZ, 1994.

Appendix 1: Training Set for Step 1

Dist.	Skewness	Quantiles Q0.125	Q0.25	Q0.375	Q0.5	Q0.625	Q0.75	Q0.875	Output					
W(3)	0.16810	0.22883	0.30780	0.37000	0.42698	0.48452	0.54890	0.63442	1	0	0	0	0	0
N(0,1)	0.00000	0.32520	0.39751	0.45158	0.50000	0.54842	0.60249	0.67480	1	0	0	0	0	0
B(3,3)	0.00000	0.25065	0.34803	0.42704	0.50000	0.57296	0.65197	0.74935	1	0	0	0	0	0
B(5,5)	0.00000	0.28121	0.36877	0.43742	0.50000	0.56258	0.63123	0.71879	1	0	0	0	0	0
B(2,2)	0.00000	0.21363	0.32173	0.41363	0.50000	0.58637	0.67827	0.78637	1	0	0	0	0	0
E(1)	2.00000	0.01750	0.03779	0.06177	0.09113	0.12898	0.18233	0.27353	0	1	0	0	0	0
W(1/2)	6.61876	0.00031	0.00143	0.00382	0.00832	0.01665	0.03326	0.07485	0	1	0	0	0	0
G(1/2)	2.82843	0.00204	0.00838	0.01972	0.03755	0.06496	0.10922	0.19425	0	1	0	0	0	0
L(3/2)	13.00000	0.00123	0.00256	0.00440	0.00713	0.01154	0.01971	0.04030	0	1	0	0	0	0
L(1)	6.18488	0.01041	0.01761	0.02572	0.03590	0.04989	0.07181	0.11641	0	1	0	0	0	0
B(.8,2)	0.77055	0.03732	0.09155	0.15789	0.23710	0.33236	0.45099	0.61269	0	0	1	0	0	0
B(1.5,5)	0.82353	0.07653	0.13210	0.18709	0.24595	0.31304	0.39596	0.51559	0	0	1	0	0	0
B(1,2)	0.56569	0.06583	0.13681	0.21402	0.29941	0.39634	0.51131	0.66115	0	0	1	0	0	0
B(1.5,3)	0.51025	0.10600	0.18031	0.25162	0.32541	0.40626	0.50123	0.62815	0	0	1	0	0	0
W(2)	0.63111	0.12545	0.18796	0.24253	0.29627	0.35399	0.42238	0.51915	0	0	1	0	0	0
G(2)	1.41421	0.05794	0.09324	0.12774	0.16519	0.20930	0.26696	0.35870	0	0	0	1	0	0
G(3)	1.15470	0.08998	0.13255	0.17155	0.21210	0.25827	0.31682	0.40718	0	0	0	1	0	0
L(1/2)	1.75019	0.07409	0.10438	0.13223	0.16175	0.19637	0.24214	0.31757	0	0	0	1	0	0
G(3)	1.15470	0.08998	0.13255	0.17155	0.21210	0.25827	0.31682	0.40718	0	0	0	1	0	0
G(2)	1.41421	0.05794	0.09324	0.12774	0.16519	0.20930	0.26696	0.35870	0	0	0	1	0	0
B(5,1.5)	-0.82353	0.48441	0.60405	0.68696	0.75405	0.81291	0.86791	0.92347	0	0	0	0	1	0
B(3,1.5)	-0.51025	0.37186	0.49877	0.59374	0.67459	0.74838	0.81969	0.89400	0	0	0	0	1	0
B(2,.8)	-0.77055	0.38732	0.54901	0.56084	0.76290	0.79137	0.90845	0.96268	0	0	0	0	1	0
B(2,1)	-0.56569	0.33885	0.48869	0.60366	0.70059	0.78598	0.86319	0.93417	0	0	0	0	1	0
B(2,.8)	-0.77055	0.38732	0.54901	0.56084	0.76290	0.79137	0.90845	0.96268	0	0	0	0	1	0
U(0,1)	0.00000	0.12500	0.25000	0.37500	0.50000	0.62500	0.75000	0.87500	0	0	0	0	0	1
U(0,1)	0.00000	0.12500	0.25000	0.37500	0.50000	0.62500	0.75000	0.87500	0	0	0	0	0	1
U(0,1)	0.00000	0.12500	0.25000	0.37500	0.50000	0.62500	0.75000	0.87500	0	0	0	0	0	1
U(0,1)	0.00000	0.12500	0.25000	0.37500	0.50000	0.62500	0.75000	0.87500	0	0	0	0	0	1
U(0,1)	0.00000	0.12500	0.25000	0.37500	0.50000	0.62500	0.75000	0.87500	0	0	0	0	0	1

Appendix 2: Training Sets for Step 2

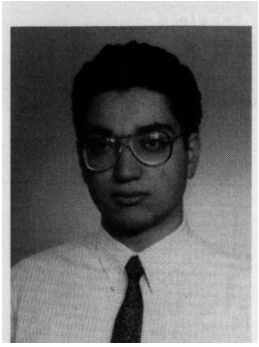
Group 1							
Dist.	**K/CV**	**Range**	**Output**				
W(3)	7.5101	1.43916	1	0	0	0	0
N(0,1)	0	4.9273	0	1	0	0	0
B(3,3)	6.1734	0.79734	0	0	1	0	0
B(5,5)	8.4191	0.67108	0	0	0	1	0
B(2,2)	4.7916	0.90358	0	0	0	0	1

Group 2							
Dist.	1/CV	Range	Output				
E(1)	1	5.3	1	0	0	0	0
W(1/2)	0.44721	37.5	0	1	0	0	0
G(1/2)	0.70711	5.06	0	0	1	0	0
L(3/2)	0.53593	55	0	0	0	1	0
L(1)	0.76287	11	0	0	0	0	1

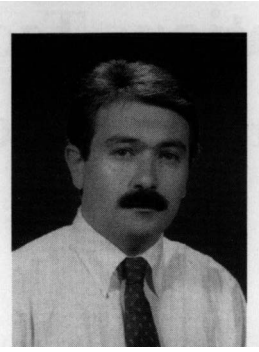
Group 3								
Dist.	Range	Skew.	Q0.125	Output				
B(.8,2)	0.9737	0.77055	0.03732	1	0	0	0	0
B(1.5,5)	0.8164	0.82353	0.07653	0	1	0	0	0
B(1,2)	0.9774	0.56569	0.06583	0	0	1	0	0
B(1.5,3)	0.936	0.51025	0.106	0	0	0	1	0
W(2)	2.7346	0.63111	0.12545	0	0	0	0	1

Group 4						
Dist.	**Range**	**Kurtos.**	**Q0.5**	**Output**		
G(2)	9.9667	6	0.16519	1	0	0
G(2)	8.97003	6	0.16519	1	0	0
G(3)	11.9017	5	0.2121	0	1	0
G(3)	10.7115	5	0.2121	0	1	0
L(1/2)	4.9894	8.89845	0.16175	0	0	1
L(1/2)	4.49046	8.89845	0.16175	0	0	1

Group 5									
Dist.	**1/CV**	**Q0.5**	**Q0.75**	**Q0.875**	**Q0.9**	**Output**			
B(5,1.5)	5	0.75405	0.86791	0.92347	0.93534	1	0	0	0
B(3,1.5)	3.31663	0.67459	0.81969	0.894	0.91018	0	1	0	0
B(2,.8)	3.08221	0.7629	0.90845	0.96268	0.97191	0	0	1	0
B(2,1)	2.82843	0.70059	0.86319	0.93417	0.94775	0	0	0	1



Anil Yilmaz is an expert at the Turkish Prime Ministry—Undersecretariat of the State Planning Organization. He received a BSc degree in Industrial Engineering and an MA degree in Economics from Bilkent University in Turkey. He has been associated with the planning of public investments, project appraisal, and investment analysis and monitoring. Yilmaz is currently working as the Counsellor to the Undersecretary.



Ihsan Sabuncuoglu is an Associate Professor of Industrial Engineering at Bilkent University. He received BS and MS degrees in Industrial Engineering from Middle East Technical University and a PhD in Industrial Engineering from Wichita State University. Dr. Sabuncuoglu teaches and conducts research in the areas of neural networks, simulation, scheduling, and manufacturing systems. He has published papers in *IIE Transactions, International Journal of Production Research, Journal of Manufacturing Systems, International Journal of Flexible Manufacturing Systems, International Journal of Computer Integrated Manufacturing, Computers and Operations Research, European Journal of Operational Research, Production Planning and Control, Journal of Operational Research Society, Computers and Industrial Engineering, International Journal of Production Economics, Journal of Intelligent Manufacturing* and *OMEGA-International Journal of Management Sciences*. He is on the Editorial Board of *Journal of Operations Management* and *International Journal of Operations and Quantitative Management*. He is an associate member of Institute of Industrial Engineering and Institute for Operations Research and Management Science.