Asian Journal of Information Technology 4 (11): 1102-1109, 2005

Exploring the Choice of Experimental Design Used to Create the Training Set for a Reverse Neural Network Simulation Metamodel in System Design

[1]Mahdi Nasereddin and Mansooreh Mollaghasemi
[1]School of Information Sciences and Technology, Pennsylvania State University,
Berks Campus Reading, Pennsylvania 19610, USA
[2]Department of Industrial Engineering and Management Systems,
University of Central Florida Orlando, Florida 32826, USA

Abstract: In this study the use of reverse simulation metamodels as a decision support tool is explored. In reverse simulation metamodeling the outputs of the simulation model (performance measures) are used as the inputs to the metamodel and the metamodel approximates the inputs of the simulation (controllable factors). The focus of this study is the choice of the experimental design (D-optimal or orthogonal arrays) used to generate the data set used to create the reverse simulation metamodel was investigated using 36 simulation scenarios with different degrees of complexity. It was found that neural network metamodels trained using an orthogonal training set performed better than those trained using a D-optimal training set.

Key words: Discrete system simulation, decision support, neural networks, experimental design, orthogonal design, D-optimal

INTRODUCTION

System design/redesign is a complex process in which models are used to make decisions on changes in existing or proposed systems. The goal of the design process is to design a system that meets or exceeds certain performance measures without violating any constraints. Simulation modeling is one of the most popular tools for the design and analysis of complex systems. This popularity is due to its flexibility, its ability to model systems more accurately and its ability to model the time dynamic behaviour of the system. With simulation modeling, however, the relationships between the design parameters and performance measures are not explicitly known. Therefore, system design using simulation becomes a trial and error process in which a set of design parameters is used in the simulation model to predict a set of performance measures. If the performance measures are acceptable, a good design has been identified; otherwise, the process is repeated until a satisfactory set of performance measures is obtained. Because of the iterative nature of this procedure, this process can be time consuming and expensive.

In order to overcome this problem, researchers have proposed the use of metamodels. A metamodel is a model of a model. The main objective of a simulation metamodel is to accurately represent the relationship between inputs and outputs over wide ranges of interest and to be more computationally efficient than simulation[1].

Two approaches have been used for developing simulation metamodels: the direct simulation metamodeling approach and the reverse simulation metamodeling approach. When building the metamodel using the direct approach, the inputs of the simulation (design parameters) are used as inputs for the metamodel and the outputs of the simulation (performance measures) are used as desired outputs for the metamodel. The direct metamodel is generally used for prediction or optimization. When used for optimization it is usually used in conjunction with an optimization technique (e.g. GA and simulated annealing). When building a reverse simulation metamodel, the outputs of the simulation (performance measures) are used as inputs for the metamodel and the inputs of the simulation (design parameters) are used as desired outputs in the metamodel. The reverse simulation metamodel is generally used to achieve specific levels of performance measures. The advantage of using a reverse simulation metamodel as a decision support tool is that the process is no longer iterative. The decision-maker inputs the required level of performance measures and the reverse metamodel outputs the necessary parameters to achieve those measures.

There has been many neural networks applications in the Simulation area. Oren[2] referenced 198 papers in the application of artificial intelligence and simulation. But most of the study was in the knowledge-based systems and simulation. Within the area of simulation and neural networks two different areas of simulation

Corresponding Author: Mahdi Nasereddin School of Information Sciences and Technology, Pennsylvania State University, Berks Campus Reading, Pennsylvania 19610, U.S.A

metamodels have surfaced in the last decade: the direct simulation metamodeling and the reverse simulation metamodeling. In this study, we will only discuss the reverse simulation metamodels. For direct simulation metamodels, the reader is referred to Kilmer, *et al*[1] and Badiru and Sieger[3]. The use of neural networks as a metamodeling technique to do the reverse of simulation modeling has been reported in three papers[4-6].

The use of optimality criteria to select the data set used in training a neural network simulation metamodel has not been explored in the literature. Several studys in the neural network literature have discussed the use of an optimal training data set to get the maximum amount of information possible. One example is the paper written by Choueiki and Mount-Campbell[7]. The authors discussed the use of the D-optimality criterion when selecting data for training neural networks.

PROBLEM STATEMENT

This study will focus on how should the training data set used to generate the metamodel be selected. The objective here is to identify an appropriate experimental design for selecting the training set that would maximize the informational content while minimizing the number of data points in the training set. By minimizing the number of data points, the number of the simulation runs needed to generate the metamodel is minimized, thus reducing the amount of time needed to generate the simulation metamodel. As mentioned earlier, Choueiki and Mount-Campbell[7] investigated the use of optimality criteria to select the training set used to train a neural network simulation metamodel. Choueiki and Mount-Campbell[7] reported encouraging results but did not compare the use of D-optimal designs to other standard experimental design techniques. Orthogonal arrays are widely used in the experimental design area[8]. In this study the performance of simulation metamodels generated using a D-optimal training set is compared to those generated using an orthogonal training data set.

EXPERIMENTAL DESIGN

In order to achieve the above study objectives, 36 simulation scenarios with different degrees of complexity were created. The 36 scenarios were generated using a full factorial design of four complexity factors which are:

- The number of reentrant levels: In a reentrant line, the product may visit the same station more than once at different stages of processing (levels). The number of processing levels had three settings: a single processing level (non reentrant model), 2 processing levels, and 4 processing levels.
- The number of machine cells required for processing: A machine cell contains several identical machines. When a product arrives to a machine cell for processing, it waits in the processing queue. When a machine is available, it is assigned a product from the processing queue based on a scheduling policy (SPT or FIFO). The number of machine cells that are required for processing the product before it exits the system had three levels: 4, 6 and 8.
- The variability within the simulation model: The variability of the simulation model was changed by varying the standard deviations of the processing times of the machine cells. The standard deviations of the processing times used were proportional to the mean of the processing times ($\sigma = k * \mu$ where k is a constant). Two levels of variation were used: The standard deviation being equal to 25% of mean ($\sigma = 0.25 * \mu$) producing a 95% confidence interval for the mean equal to ($\mu \pm 2*(0.25 * \mu) = (0.5, 1.5\ \mu)$) and 10% of the mean ($\sigma = 0.1 * \mu$) producing a 95% confidence interval of the mean equal to ($\mu \pm 2*(0.1 * \mu) = (0.8$ and $1.2\ \mu)$).
- System saturation: This is measured by dividing the mean arrival time by the total mean processing time of the bottleneck. A saturation value close to 1 would cause the queue times to grow exponentially The smaller the value of the saturation coefficient the smaller the queue times will be. This will lead to smaller cycle times, but at the same time produces smaller system throughput. Thus there is a trade off between cycle times and throughput. Two saturation levels were used (0.9 and 0.7).

In order to accomplish the study objectives, a full factorial design was developed using 5 factors, 4 of which were the complexity factors mentioned earlier leading to 36 different scenarios and the other factor was the type of experimental design used to generate the training data set: Two experimental designs were used: D-optimal design and orthogonal arrays design. D-optimal designs provide great flexibility in terms of the number of data points used to generate the design. These are the only experimental designs used in the literature where the user can specify the exact number of data points to be chosen from a list of candidate experimental data points. D-optimal designs provide great flexibility while attempting to maximize the informational contents of the data by minimizing correlation. An orthogonal arrays design is less flexible than a D-optimal design in terms of the number of data points and the ability to select specific points to be part

of the experimental design, but it provides an optimal design in terms of minimizing correlation[9]. The full factorial design above results in 72 different simulation metamodels (2x36). The experimental design used is summarized in Table 1.

- The simulation models: Thirty-six different reentrant simulation models with different complexity were built using the Arena simulation software package. Depending on the number of machine cells (4, 6, or 8) the simulation models had 5, 7, or 9 controllable factors (Table 2). Thus the 4 machine cells models had a solution space of 162 configurations each, the 6 machine cells models had a solution space of 1458 configurations each, and the 8 machine cells models had a solution space of 8748 configurations each. Thus if a decision-maker requests a specific set of performance measures to be achieved, many simulation runs will be needed to find the design that meets his requirements.

The simulation inputs are:

- The number of machines in each of the machine cells:
- The scheduling policy: When a product finishes processing at a specific machine, it is routed to the next machine cell for processing. It enters the machine cell queue and waits its turn for processing. When a machine is available in the machine cell, a product in the machine cell queue is chosen for processing using a scheduling policy. Two scheduling policies were used (First In First Out (FIFO) or Shortest Processing Time (SPT)).

The simulation outputs are:

- The average utilization of the machines in each of the machine cells (4, 6, or 8 outputs).
- The average work in progress (WIP).
- The average cycle time.

Because variability exists in the system, the value of the inter-arrival time should be less than the total of the processing time of the bottleneck. For the purpose of experimentation, two saturation levels have been used (0.9 and 0.7). In order to achieve saturation levels of 0.7 and 0.9, the following steps were taken:

- The total processing time over all processing levels per machine was calculated.
- The bottleneck was identified. The bottleneck is the machine that restricts the capacity of the system, thus the machine with the highest total mean processing time was identified as the bottleneck of the system.
- The inter-arrival time value was calculated. If the value of the inter-arrival time is smaller than the total processing time of the bottleneck, then the system will be over-saturated and the number of entities in queue will keep on increasing.

$$\text{Mean inter-arrival time} = \frac{\text{(The total of the mean processing times of the bottleneck)}}{\text{(The saturation level)}}$$

In order to get a statistically valid estimate of the mean for each of the simulation outputs several replications of the simulation model were required. Also in order to remove the bias caused by starting with an empty model, a warm up period was needed. In order to establish the number of simulation replications, runs were made until the half width of all outputs across replications were less than or equal to 10% of the average output[10]. In order to remove the initialization bias of the system, a warm-up period was identified. In order to establish the warm-up period, a base model (the model with 8 machines and 4 levels) was run for 100,000 time units. The work in progress versus time chart and the cycle time versus time chart were analyzed. It was established that a warm-up period of 2,000 time units is needed to remove the initialization bias.

The models above were used in studying the effect on the metamodel prediction accuracy of the type of the experimental design used in identifying the training set.

Table 1: Experimental design used to test the effect of various factors on the performance of a simulation metamodel

Factors	No. of levels	Levels
The experimental designs used to generate the training data	2	D-Optimal Orthogonal array
The number of re-entrant levels	3	1, 2 and 4
The number of machine cells	3	4, 6 and 8
Variability in the model	2	Standard deviation of the processing time 10 and 25% of the mean processing time
System saturation	2	0.7 and 0.9
Total	72 Combinations	

Table 2: The simulation models inputs and solution space

Simulation input	No. of different settings(4 Machine cells models)	No. of different settings (6 Machine cells models)	No. of different settings (8 Machine cells models)
Cell 1	3	3	2
Cell 2	3	3	3
Cell 3	3	3	3
Cell 4	3	3	3
Cell 5	-	3	3
Cell 6	-	3	3
Cell 7	-	-	3
Cell 8	-	-	3
Scheduling policy	2	2	2
No. of possible solutions	162	1458	8748

The D-optimal and orthogonal arrays designs used to generate the training data: When building simulation metamodels, many simulation runs are needed to generate the required data. Usually these simulation runs are expensive and time consuming. So it is important to minimize the number of simulation runs necessary to generate the metamodels. The objective of the design of experiment is to extract the maximum amount of unbiased information regarding the factors affecting a response from as few observations as possible.

Because the simulation models used in this research have three different solution space sizes (162, 1458 and 8747), three orthogonal designs and three D-optimal designs were needed for experimentation. The orthogonal designs (Table 3) were generated using the JMP statistical package. JMP uses standard orthogonal designs. The D-optimal designs (Table 4) were generated using JMP's DETMAX algorithm implementation. Because of the local maximum problem suffered by the D-optimal design, DETMAX was run 10 times using random starting points to minimize the effect of the local maximum problem. The best runs (the ones with the highest D-efficiency value) for all the D-optimal models were used. None of the 3 D-optimal designs had a 100% D-efficiency, but all were higher than 98%, indicating that some linear dependency exists between the data points. The covariance matrix for the D-optimal designs was calculated.

Training the neural network metamodel using back-propagation: A feed-forward neural network was trained using the delta rule. The network used consisted of three layers: an input layer, a hidden layer, and an output layer. The choice of using a three-layered network rather than a four-layered network was based on recommendations made by Villiers and Banrnard[11]. An F-offset of 0.1 was used to prevent network saturation (large weights, large summation values and learning difficulty). The parameters of the neural network are summarized in Table 5.

Table 3: Orthogonal arrays designs used

Design No.	No. of factors	No. of points	Resolution
1	9 Total 8 Machine cells scheduling policy	36	III
2	7 Total 6 Machine cells scheduling policy	18	III
3	5 Total 4 Machine cells scheduling policy	18	III

Table 4: D-optimal designs used

Design No.	No. of factors	No. of points	D-efficiency	A-efficiency	G-efficiency
1	9 Total 8 Machine cells scheduling policy	36	99.88	99.75	98.77
2	7 Total 6 Machine cells scheduling policy	18	98.37	96.97	94.28
3	5 Total 4 Machine cells scheduling policy	18	98.84	97.78	95.74

Table 5: The back-propagation neural network parameters

Parameters	Value
Activation function	Sigmoidal
Number of hidden layers	1
Number of hidden nodes	4
Learning coefficient (Hidden layer)	0.3
Learning coefficient (Output layer)	0.15
F-offset	0.1

MATERIALS AND METHODS

In order to achieve the study objectives, the following steps were taken (Fig. 1)

- In order to test if the complexity of the re-entrant simulation model affects the prediction capability of the reverse neural network simulation metamodel, thirty-six re-entrant simulation problems with different complexity were generated. The complexity of the problems was defined as a function of four factors:

  Number of machine cells required for processing
  Number of re-entrant levels required for processing
  Variability within the simulation model.
  Saturation of the simulation model.

- To test if the type of experimental design used to generate the training data affects the performance of the simulation metamodel, two training data sets were generated using the D-optimal and orthogonal arrays experimental design for each of the re-entrant simulation problems.
- In order to test the prediction capabilities of the neural network metamodels, test data sets that covered the whole solution space were generated using a full factorial design of all possible number of machines per machine cell for all machine cells and scheduling policies. Thus for the 4 machine cells models, 162 different test points were generated. For the 6 machine cells models, 1458 different test points were generated. For the 8 machine cells models, 8748 different test points were generated. Thus a total of 124,416 simulation runs were needed to generate the test data sets.
- The test data was run through the trained reverse neural networks simulation metamodels, and the error of prediction for each of the simulation metamodels was calculated (The error of prediction = Expected Neural Network Output – Actual Neural Network Output).

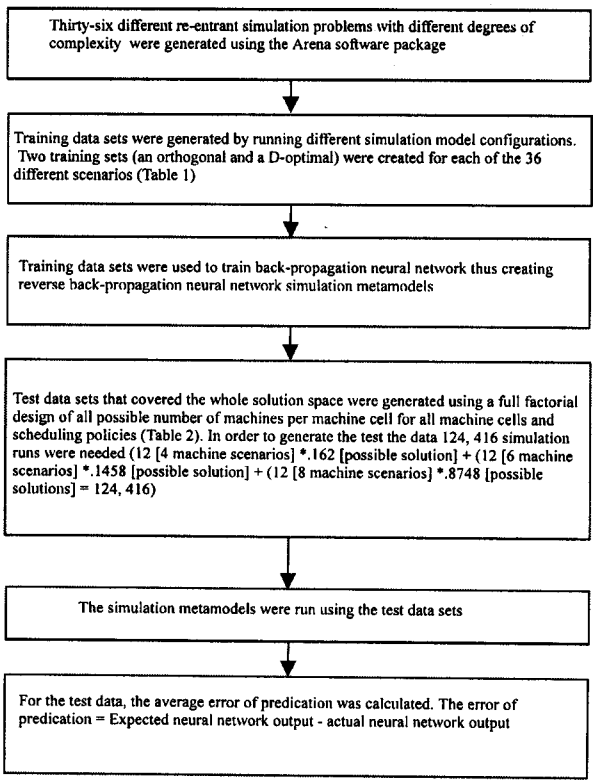

Fig. 1: The research methodology summary

- After analysing the results the following questions would be answered:
  - How does complexity (as defined by its four factors) affect the performance of the neural network metamodel?
  - Should D-optimal or orthogonal arrays be used to generate the training set used to train the neural network metamodel?

RESULTS

Two back-propagation neural network simulation metamodels were created for each of the 36 different simulation scenarios. One metamodel was created using a D-optimal training set, while the other was created using an orthogonal array training set. Test data sets were then used to measure the performance of the neural network metamodel in terms of its prediction capability. For each scenario, a test data set was generated using a full factorial design as described in Table 2. For example, for a 4-machine cells scenario, 162 simulation configurations (full factorial design) were generated. For the 36 test data sets, a total of 124,416 simulation scenarios were run (12 [4 machine scenarios] *.162 [possible solutions] + (12 [6 machine scenarios] *.1458 [possible solutions] +(12 [8 machine scenarios] *.8748 [possible solutions] = 124,416).

After running the test data set through the back-propagation neural networks metamodel, the output of the neural network was compared to the expected output (simulation input). For each input-output pair, if the metamodel correctly identified all the simulation inputs then a correct classification is noted, otherwise a misclassification is noted. Then the percent misclassification for the entire test set is calculated.

To analyse the data, a stepwise regression model was created with the percent misclassifications as the dependent variable (response), and the variability, saturation, number of machines, number of levels and the experimental design used to generate training data as the independent variables (predictors). All second order interactions of the predictors were also used in the model. The ANOVA table generated by the regression model is shown in Table 6. The relationship between the response (% misclassification) and the predictors was found to be:

Table 6: Analysis of Variance (R^2 = 68% and R^2 adjusted = 64%)

Source	DF	SS	MS	F	P
Regression	8	16507.3	2063.4	16.92	0.000
Residual Error	63	7684.0	122.0		
Total	71	24292.4			

$$\text{Misclassification \%} = -74.3 + 266\,A + 31.5\,C - 2.92\,D + 45.7\,E - 2.51\,C^2 - 287\,AB - 30.1\,AE - 36.2\,BE \text{ (All significant at } \alpha = 0.1)$$

Where:

A is the variability of the simulation model (0.1 and 0.25)

B is saturation of the simulation model (0.7 and 0.9)

C is the number of machines in the simulation model (4, 6 and 8)

D is the number of levels (1, 2 and 4)

E is the coded experimental design used to generate the training data (-1 for orthogonal arrays and 1 for D-optimal)

From the regression analysis the following conclusions can be made:

- Neural networks that use orthogonal arrays to select the training data set performed better than neural networks that use D-optimal design (at least for the manufacturing simulation metamodeling area) (Fig. 2). An important question that we need to answer is why orthogonal arrays outperformed D-optimal designs? Orthogonal arrays by design have zero correlation within the data set. A D-optimal design with a D-efficiency of 100% has zero correlation. In all of our experiments the DETMAX algorithm failed to generate a D-optimal design with a D-efficiency of 100%. Thus all the D-optimal designs that were used were not correlation free. Thus the D-optimal designs used did not maximize the amount of information that could have been extracted with the limited amount of data points. To further examine the effect of correlation on the experimental design efficiency, the correlation matrix for all the training data sets (3 D-optimal and 3 orthogonal) was generated. For the orthogonal training data sets, it was found that the correlation matrix is a diagonal matrix thus indicating zero correlation between the factors. As for the D-optimal training sets, it was found that weak correlation exists between some of the factors in the 4 and 6 machine cells design. The question is: Could weak correlation have such an impact on the performance of the metamodel? In answering this question it is important to realize how small the training set is in comparison with the possible solution space. I believe that because of this fact that even small correlation will affect the performance of the metamodel. More experimentation is needed.
- The saturation of the simulation model had minimal effect on the performance of the back-propagation neural network. This might be caused by the fact that our experimental settings values were too close (0.7 and 0.9). Thus more experimentation is needed to establish the effect that saturation has on the performance of the back-propagation neural network metamodel.
- Variability within the simulation model significantly affected the performance of the neural network metamodel. As the variability within the simulation model increased, the % misclassification of the metamodel increased (Fig. 3). In other words, the higher the variability within the simulation model, the harder it is for the metamodel to identify the correct solution. This is expected since the more variability within the system, the more complex the system is thus the harder it is for the neural network metamodel to predict the simulation output.
- As the number of machine cells increases, the % misclassification of the metamodel decreased (Fig. 4). In other words, the lower the number of machines, the harder it was for the metamodel to identify the correct solution. This is an unexpected result since the more machine cells within the system, the more complex the system is thus the harder it should be for the neural network metamodel to predict the simulation output. A closer look reveals that there is another factor that causes this to occur which is the fact that the efficiency of the experimental design used to generate the training data sets is dependent on the number of machine cells in the system. The neural networks generated using orthogonal training data sets all have equal correlation (correlation =0), but the neural networks generated using D-optimal training data sets have different correlation values depending on the number of machine cells in the system (the 8 machine cells had the highest D-efficiency (99.88%), the 6 machine cells had the lowest D-efficiency (98.37%), and the 4 machine cells had a D-efficiency of 98.84%). Thus for the D-optimal results, the efficiency of the experimental design used to generate the training data sets is confounded with the number of machine cells in the system. So if we look at the orthogonal results which has equal experimental design efficiency, we will be able to have a clearer representation of how the number of machine cells affect the performance of the neural

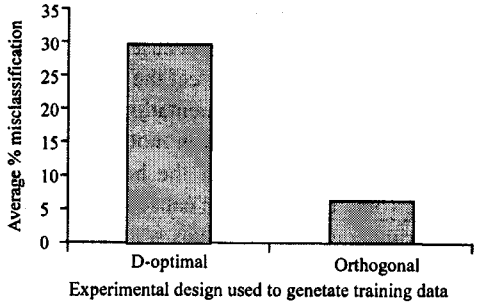

Fig. 2: Average misclassification based on the experimental design used to generate training data

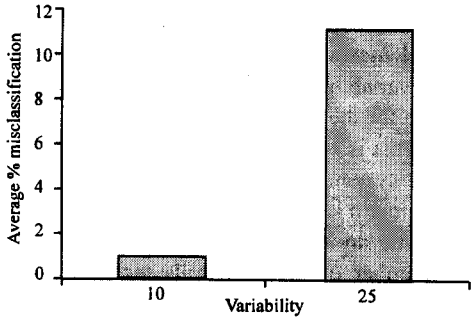

Fig. 3: Average variability misclassification rate

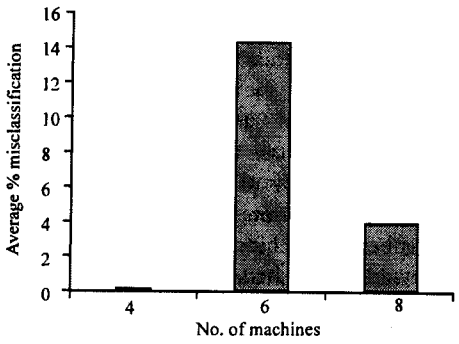

Fig. 4: Average number of machines misclassification rate

variability within the simulation model. All the other factors (Fig. 5 and 6) were statistically insignificant network metamodel. A regression analysis was performed using only the results from the neural networks that were generated using orthogonal experimental design. It was found that the number of machine cells in the system does not affect the performance of the neural network metamodel (Table 7). Actually the only significant factor was the

Table 7: The regression analysis results using only neural networks that were generated using orthogonal arrays

	Coefficients	S E	t Stat	P-value
Intercept	-0.0720	0.118	-0.610	0.545
Variability	0.0066	0.003	2.06	0.048
Number of machines	0.0099	0.015	0.661	0.512
Number of levels	-0.0186	0.020	-0.950	0.349

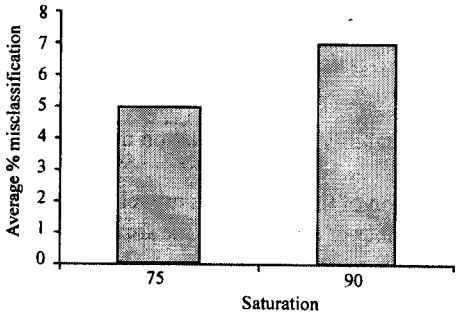

Fig. 5: Average saturation misclassification rate

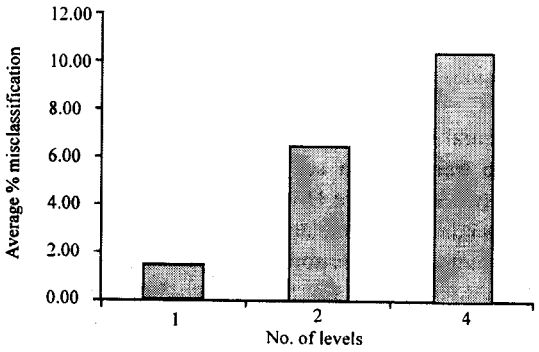

Fig. 6: Average number of levels misclassification rate

(at $\alpha = 0.1$). It is important to understand that the reason the saturation of the simulation model was insignificant might have been caused by the experimental settings used (0.7 and 0.9). More experimentation is needed to determine the effects saturation has on the performance of the neural network metamodel.

CONCLUSIONS

In this study the choice of the experimental design used to generate the training data was investigated. For the 36 scenarios tested, it was found that orthogonal arrays designs perform better than D-optimal designs in terms of providing the neural network with the maximum amount of information given the smallest number of data points. So, in order to maximize the informational contents

of a data set, it is recommended to use orthogonal experimental design to create the neural network training set. Does that mean that D-optimal designs should not be used to create training sets for neural network metamodels? The answer is situation dependent. D-optimal designs provide greater flexibility than orthogonal arrays. When using D-optimal designs, the designer can specify the list of possible design points. The designer can include data points that were performed in previous experimentation, and can also exclude data points that are too computationally expensive. Also orthogonal arrays are limited by a pre-set number of data points that are required in a design. When using D-optimal the number of data points can be specified by the designer thus giving the designer greater flexibility.

FUTURE STUDY

One of the major shortcomings of back-propagation is its inability to perform a global search. Global search techniques (e.g., evolutionary algorithms, simulated annealing, and tabu search) could be used to train the neural network. We are currently exploring the use of evolutionary neural networks as reverse simulation metamodel.

The DETMAX algorithm failed to produce optimal D-optimal designs (100% D-optimal efficiency). The problem with the DETMAX algorithm is that it performs local searches (versus global searches). Other techniques (e.g., GA and simulated annealing) could be used to search for optimal D-optimal designs. Also a hybrid of a GA and the DETMAX algorithm might produce superior results. The advantage of using a hybrid is that the GA will search globally and the DETMAX algorithm will perform a local search. We are currently exploring the use of simulated annealing to produce better D-optimal designs.

The effect of D-efficiency on the performance of neural networks trained using D-optimal designs should be investigated. In some cases, the use of orthogonal array designs is infeasible, thus D-optimal designs become the only designs available. Thus there is a need to understand the relationship between D-efficiency and the performance of the neural network. This will give the analyst a gauge of the performance of the neural network metamodel

REFERENCES

1. Kilmer, A.R., S. Alice and L.J. Shuman, 1997. An Emergency, J. Soc. Health Sys., 5: 63-79.
2. Oren and Tuncer, 1994. Artificial Intelligence in Simulation. Annal. Operations Res., 53: 287-319.
3. Badiru, B. Adedeji and B. Sieger David, 1997. Neural networks as a simulation metamodel in economic analysis of risky projects. Eu. J. Operations Res., pp: 1-13.
4. Mollaghasemi, Lecroy and Georgiopoulos, 1998. Application of neural networks and simulation modeling in manufacturing system design. Interfaces, 5: 100-114.
5. Chryssolouris, G., M. Lee, J. Pierce and M. Domroese, 1990. Use of neural networks for the design of manufacturing systems. Am. Soc. Manufacturing Eng., 3: 187-194.
6. Chryssolouris, G. and M. Domroese, 1991. The Use of neural networks in determining operational policies for manufacturing systems. J. Manufacturing Sys., 10: 166-175.
7. Choueiki, M. Hisham, Mount-Cambell and A. Clark, 1999. Training data development with the D-optimality criterion. IEEE Transactions on Neural Networks, 10: 56-63.
8. Myers, R.H. and D.C. Montgomery, 1995. Response Surface Methodology. John Wiley and Sons Inc., New York.
9. Croarkin, C. and P. Tobias, 2000. Engineering Statistics Handbook, (The National Institute of Standards and Technology) [www http://www.itl.nist.gov/div898/handbook /pri/section5/pri521.htm].
10. David, K., S. Randall and S. Deborah, 1998. Simulation and Arena. McGraw-Hill Publishing Company, New York.
11. Villiers Jacques de and B. Etienne, 1992. Backpropagation Neural Nets with One and Two Hidden Layers IEEE Transactions on Neural Networks, 4: 136-14.