Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, eds

ALLOCATION OF SIMULATION EFFORT FOR NEURAL NETWORK VS. REGRESSION METAMODELS

Corinne MacDonald
Eldon A. Gunn

Dalhousie University
PO Box 15000
Halifax, NS B3H 4R2, CANADA

ABSTRACT

The construction of a neural network simulation metamodel requires the generation of training data; design points (inputs) and the estimate of the corresponding output generated by the simulation model. A common methodology is to focus some simulation effort in obtaining accurate estimates of the expected output values by executing several simulation replications at each point and taking the average as the estimate. However, with a limited amount of simulation effort available and a rather large input space, this approach may not produce the best expected value approximations. An alternate approach is to distribute that same simulation effort over a larger sample of input points, even if it means the resulting estimates of the expected outputs at each point will be less accurate. We will show through several examples that this approach may result in better neural network metamodels; this conclusion differs from other studies involving regression metamodels.

1 INTRODUCTION

Simulation metamodels are deterministic mathematical functions that approximate the functional relationship between simulation variables and the expected output of a replication of the simulation model. Metamodels come in many forms, such as regression models, kriging (e.g. Kleijnen 2005), splines (e.g. Barton 1998), and neural networks (e.g. Fonseca et al. 2003). The process for constructing a metamodel involves the use of a dataset of a selection of parameter settings (design points), and the corresponding performance estimate for each design point, obtained using a simulation model. Metamodel construction involves finding a mathematical function that best "fits" this data. This deterministic function is an approximation of the expected value function that is more convenient to use than the simulation.

With regression metamodels, the form of the function must be selected prior to attempting to fit the data. If the form is linear in the model parameters, then the best fit can be achieved using a least squares approach. If the form is nonlinear, a non-exact approach must be used. In either case, the model itself may be selecting by observing the data to be fit, or with some a priori knowledge. If using a feedforward neural network, a decision on the form is not required, only the architecture of the neural network itself (number of hidden layers and nodes per layer).

Obviously, the quality of the dataset itself will impact the accuracy of the metamodel. Therefore, there have been several papers in the literature addressing different issues that arise when constructing the dataset; the number of design points required, the experimental design method used to select them, and the number of simulation replications required at each design point. However, most of this work involves using regression metamodels.

978-1-4673-4781-5/12/$31.00 ©2012 IEEE

We will show in this paper that the conclusions drawn for regression metamodels are not necessarily true for feedforward neural network metamodels. In particular, for neural network metamodels, using more design points with single observations can provide more information during training than the generally accepted practice of allocating some simulation effort to replicating at a single design point. This primarily has to do with two issues. First, the need for statistical information on the data when constructing regression metamodels requires there be some replication at each design point. Secondly, since neural networks can be viewed as nonparametric regression, they require more information across the input space in order to determine their form.

In Section 2, the issues involved in constructing simulation metamodels are described, including how such issues are not the same for regression and neural network metamodels when it comes to allocating simulation effort. In Section 3, we present several examples, some taken from the literature, that demonstrate why allocating simulation effort to exploring more design points can lead to better neural network metamodels. Our discussion and conclusions are in Section 4.

2 CONSTRUCTING SIMULATION METAMODELS

2.1 Issues in Metamodel Construction

Each type of simulation metamodel has its own considerations during metamodel construction. A function approximation metamodel must be able to *generalize*; that is, to produce reasonable estimates for input not included in the dataset used during its construction. For regression models, choosing a more complex model form than is necessary (over-parameterization) could mean developing a model that fits the training data closely, but does not generalize well. In neural network metamodeling, the approach typically used is to keep the architecture of the network as simple as possible or regularization by employing some type of penalty term in the error function that penalizes overly complex models (e.g. Saito and Nakano 2000). While this introduces some bias into the fitting error measure, it also can reduce error due to variance (the "Bias/Variance Dilemma" discussed by Geman et al. 1992). A related issue in the neural network literature is referred to as overtraining or overfitting (Smith 1993), where, in the attempt to minimize the overall training error, the model is fit too closely to the training data and does not generalize well on new data. This is a common issue for networks trained with sparse training datasets, regardless of the accuracy of the training data or the size of the network, because neural networks typically have a relatively large number of degrees of freedom in the weights to estimated. A common approach to avoiding such overfitting is to monitor both the error measure for the training data and a test dataset during training. The training is then stopped when the error begins to increase for the test dataset. Taking such an approach, even with ample training data, can improve the ability of an overparameterized model to generalize by preventing it from developing to its full complexity (Sjöberg and Ljung, 1995). Either early stopping or regularization may be a good approach when the function to be approximated is known to be a relatively smooth, monotonic function, considered to be "well behaved" over the input space. If it is known to be a complex function, or if no information is known, then such an approach may not be appropriate.

2.2 Generating Simulation Responses

It is assumed that the simulation model used to generate observations for the selected design points is well designed and that its goal is to estimate the steady state expected value of at least one output. It is also assumed that the simulation provides an unbiased estimate of the expected value of the output, although achieving unbiased estimates of an expected value via simulation is far from trivial. At the very least, this involves a judicious choice of a warmup period and a data collection period that is sufficiently long (Law and Kelton, 2000). Obviously very long sample paths of an unbiased simulation can produce very accurate estimates of the output expected value; obtaining multiple observations of the same scenario (replications) and calculating the mean value should also result in more accurate estimates. Such accuracy is re-

quired when the goal is to compare the outcomes of two or more specific scenarios. However, either of these approaches may be an excessive expenditure of effort for one design point when the goal is to understand the form of the expected value function over all scenarios.

2.3 Experimental Design

Kleijnen (2007) discusses the Design and Analysis of Simulation Experiments (DASE), which includes all decisions made in the design of the simulation experiments, for the purpose of building regression metamodels. Fonseca et al. (2003) note the importance of having enough information in the training set to train a neural network, but they do not address this issue further. As Kleijnen et al. (2005) note, the decision on the type of experimental design methodology depends to some extent on the ultimate use of the metamodel. For our purposes, we assume that we are dealing with a multi-dimensional input space, so that visualizing the appropriate metamodel is not possible. We also assume that we don't have a clear understanding of the shape of the response surface over the input space, and we do not know if it can be assumed to be well behaved. Therefore, to construct neural network metamodels under these circumstances, a space-filling experimental design is vital, as it provides more information about the shape of the response surface, and is the best way of exploring surfaces that are not expected to be smooth (Kleijnen et al. 2005). There are several possible approaches used for selecting the scenarios, such as Central Composite Design, Random Sampling, Full or Partial Factorial Design, and Latin Hypercube; see Alam et al. (2004) for a comparison.

2.4 Constructing the Dataset: Replication vs. Adding More Design Points

The choice of experimental design will of course be influenced by the amount of effort required to collect one simulation response, and the number of input variables involved. For the simple single input variable examples shown later in this paper, the computational expense of generating more data for the training dataset is relatively small, and therefore the argument could be made to generate both more design points and to replicate each several times. However, this is not true for the types of models that inspired this work – those with 10 or more input variables. For such a system, even a relatively poor two-factorial experimental design with 10 replications per design point would require 10,240 simulation runs. Three-factor factorial designs with 10 replications would require 590,490 runs. Depending on the complexity of the model, this may not be possible. For the generation of regression metamodels, some replication is necessary in order to use common statistical approaches to fit the data and evaluate the goodness of the fit (Santos and Santos, 2008). However, there is no such requirement for constructing neural network models.

Santos and Santos (2009a, 2009b) looked at this issue with respect to the construction of regression models. In Santos and Santos (2009a) they fitted both a linear and a nonlinear regression model (both with only a single input variable) using design points with only one observation per point. However, they first "smoothed" the responses over the responses of nearby design points before attempting to fit the model. They concluded that this was a better approach than using the same amount of simulation effort replicating fewer design points. In both cases, the functional form of the metamodels was selected based on observation of a graph of the simulation responses. In Santos and Santos (2009b), they conducted a series of experiments where they attempt to fit a second order polynomial to the time-in-queue function for an M/M/1 system as a function of the arrival rate, λ. They generated five different sets of simulation data each requiring nearly the same amount of simulation effort but with a different number of design points and replications; this included a dataset with only six design points and the observations for each averaged over 20 replications, and one with 101 equally spaced design points each sampled only once. They note that in their experiments the accuracy of the resulting regression metamodels does not seem to differ much when using more less accurate points than fewer, more accurate points. However, they conclude "that it is preferable to have more replications and less design points, since it is possible to collect more meaningful statistical information without sacrificing the accuracy of the metamodel. The number

of design points should be kept low, as far as it does not compromise the sampling of the simulation model response detail." (Santos and Santos, 2009b).

There has been little work in addressing this question when constructing neural network simulation metamodels. Kilmer and Smith (1993) used the simulation of an inventory system from Law and Kelton (2000) to compare the use of neural networks as simulation models to that of regression. Using the same 36 input scenarios, they created two separate datasets. The first set contained the observations of total cost for 10 separate observations (individual replications) of the simulation for each scenario (for a total of 360 points), while the second dataset contained only the mean of the observations as a single point for each scenario. They concluded that there was not much difference in the accuracy of the networks trained with these datasets, and both networks outperformed the second order regression models constructed from the same data. Kilmer et al. (1999) report on further experiments that show the individual replication training approach resulted in more accurate networks except for the largest set of scenarios. Hurrion and Burgil (1999) compared full-factorial and random experimental design methods, and concluded that the random method is best; for their experiments they simulated each scenario 20 times and used the mean observation in the training dataset. They also tested the idea that more accurate data could result in more accurate networks. They compared their original networks to networks trained with datasets generated from 40 and 60 replications, and found that these were more accurate, especially so for the random experimental design; the number of scenarios in both cases was also large. They did not experiment with a dataset with twice or three times as many scenarios. Generally, it seems to be accepted practice that regardless of what type of experimental design approach is used to select the scenarios for the dataset, some amount of simulation replication at each scenario is conducted, as is done for regression metamodels.

While this conclusion seems valid for the construction of regression metamodels, we show here that it may not be valid for neural network metamodels. For neural network training, statistical information which requires replication is not required on the individual points for the purposes of network training. Additionally, the absence of a pre-specified form for the metamodel function may require more information from across the input space. The following experiments show that, as long as the simulations produce an unbiased sample for each scenario, it may be better to use simulation effort to explore more scenarios rather than to obtain accurate estimates at each scenario, when generating training datasets to build neural network metamodels.

3 ILLUSTRATIVE EXAMPLES

3.1 Single Variable Example: M/M/1 Model

The first example is the same M/M/1 queueing model found in Santos and Santos (2009b). The server processes arrivals at a rate of $\mu = 1$ per minute (exponentially distributed); customers arrive according to a Poisson arrival process, with the arrival rate, λ, varied between 0.8 and 0.9 arrivals per minute (high utilization model). Thus, the utilization factor $\rho = \lambda / \mu = \lambda$. The goal was to build a metamodel for the expected time a customer spends in the queue for the server, W_q, as a function of the utilization factor, ρ. From queueing theory (e.g. Ross, 2006), we know that the expected steady state function is $E[W_q] = \rho/(1-\rho)$, a smooth, monotonically increasing function. It is also known that observations of this measure, at higher values of ρ such as studied here, are highly variable. The system was simulated using a custom built simulation model and the average time spent in the queue was measured. In Santos and Santos (2009b), the first 6000 observations were considered the warmup period and were deleted, and the next 3000 observations were collected for each simulation run. Since the simulation model used here required the specification of warmup and run length in time units, this was approximated by setting the warmup time equal to $6000/\lambda$ and the overall run time equal to $9000/\lambda$.

A set of 101 equally spaced design points were selected between 0.8 and 0.9, and each was simulated 20 times. Two sets of training data were then generated from the data. The first set consisted of only six design points (0.8, 0.82, 0.84, 0.86, 0.88 and 0.90), with the observation of time in system for each being

the average of 20 replications. The second set consisted of all 101 design points, but using only a single observation for time in system.

Obviously, the actual function would not normally be known when fitting a metamodels, but to fairly measure the accuracy of these metamodels it only makes sense to compare the metamodels output to this known function. Therefore, to compute the actual mean squared error, MSE_a, we use the expected value of W_q from this function compared to the result provided by the metamodels:

$$MSE_a = \frac{1}{n}\sum_{i=1}^{n}(z(\rho_i) - E[W(\rho_i)])^2$$

where $z(\rho_i)$ is the metamodel estimate of the expected value of W_q, n = 101 is the number of test points, and $\rho_i \in \{0.8, 0.801, \ldots, 0.899, 0.9\}$.

For the first trial, six equally spaced design points, $\rho_i \in \{0.8, 0.82, 0.84, 0.86\ 0.88, 0.9\}$, were selected, and the value of the wait time in queue was computed using the queuing function; therefore, the data could be considered exact. With the limited amount of complexity, a two hidden node network was sufficient to train the network to a reasonable degree of accuracy, both when using regularization (with ratio = 0.99) and without. The resulting metamodels are shown in Figure 1, along with the best fitting 2nd order regression metamodel.

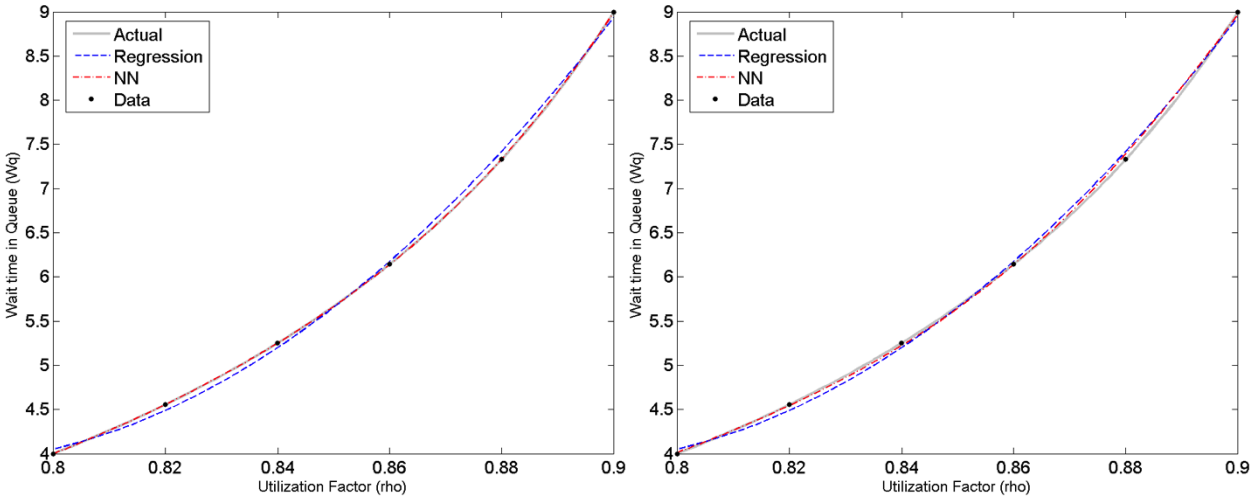

Figure 1: Neural Network metamodels with 2 hidden nodes fit to 6 actual data points; on the left, with no regularization (*MSEa* = 1.5903E-05), and on the right, using regularization (*MSEa* =9.9134E-04)

For the second trial, again the same 6 design points were used, but this time the observations are the average of 20 replications of the simulation model (discussed above). The resulting metamodels are shown in Figure 2. While there isn't much difference in the error term, there is significant difference in the shape of the two models. The regularization technique can prevent the model from developing large weights, which will produce a complex metamodel; thus a smooth function results, very close to that of the regression model. However, without regularization, the metamodel tends to do well in the lower range of the input values but is substantially different as ρ approaches 0.9 (Figure 2a). One important thing that is worth noting is that the neural network metamodel trained without regularization has fit the 6 data points very closely; there is no mechanism to keep the curve smooth, as with regularization. There is no such concern with regression models; by selecting a second order model, the general shape of the metamodel is limited, and fitting that closely to the data is not a concern.

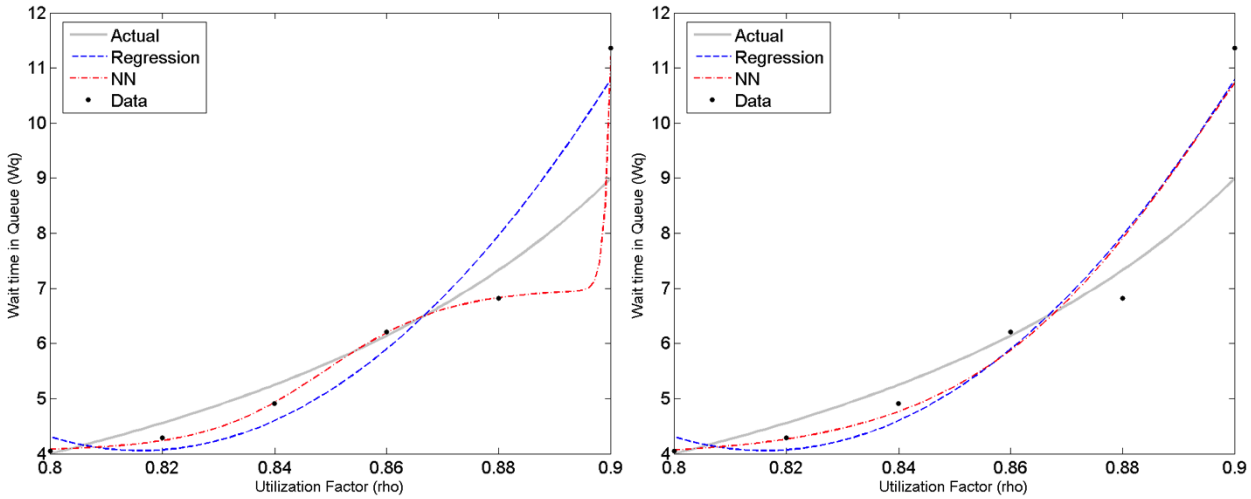

Figure 2: Neural network metamodels with 2 hidden nodes fit to 6 Data Points, averaged over 20 replications: on the left, without regularization (MSEa = 3.4572E-01), and on the right, using regularization (MSEa = 3.9842E-01)

The final set of trials uses a dataset with all 101 design points, sampled once, which involves an amount of simulation effort slightly less than that of the previous model. As can be seen in Figure 3, there is substantial amount of noise in the observations; note that the average of the 3000 steady-state observations of W_q for $\rho = 0.86$ was 17.3, where the expected value was 5. This time, the network appears to fit the data rather well, and is very similar again to the regression model. Regularization does not appear to have much of an impact on the resulting model.

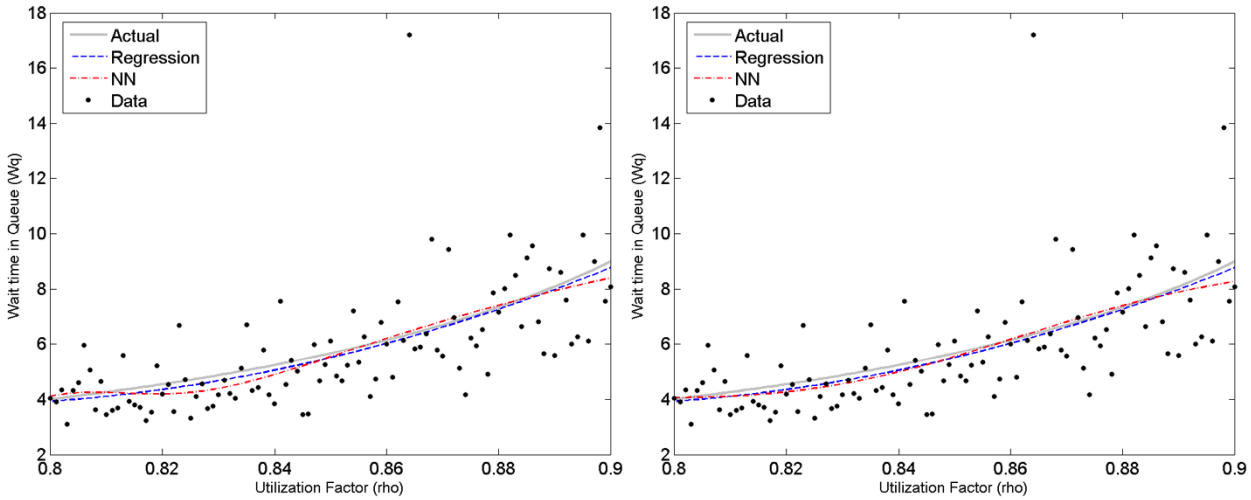

Figure 3: Neural network metamodels with 2 hidden nodes fit to 101 data points (De) (single observation per point): on the left, without regularization (*MSEa* = 6.9477E-02), and on the right, using regularization (*MSEa* = 5.4681E-02).

This example shows that without regularization, neural networks trained on smaller, more accurate (but not exact) datasets may not generalize well over the entire input space (Figure 2a); however, even without using any form of regularization, the network trained on the larger noisy dataset achieved a much improved level of accuracy.

3.2 Single Variable Example with Complexity: Sine Wave

When the underlying function is unknown, or known to be complex, we may not want to limit the ability of the neural network to take on larger weights (and hence, more complexity). In this simple example, the function to be approximated is $y = 0.5 + 0.25\sin(3\pi x)$ for $x \in [-1,1]$. Observations at any point x were generated by computing y and then adding random noise, ε, where ε was normally distributed with $E[\varepsilon] = 0$ and standard deviation of 0.1. Seven equally spaced design points were selected, and 20 samples for each were generated (requiring a total of 140 samples), and the averages computed. Those points, and a network trained on those points is shown in Figure 4. Obviously insufficient information was provided to adequately train the network. A second dataset included 13 equally spaced design points. By observing the actual function, this is about the minimum number required to build an accurate metamodel. Again, the average of 20 observations of each point were computed for the dataset. A third set of data was generated by selecting 129 equally spaced design points, and producing one noisy observation for each. The trained networks (each with 6 hidden nodes) for both attained roughly the same level of accuracy (Figure 5), yet the third required half of the observations of the second model, and about the same level of simulation effort as the first.

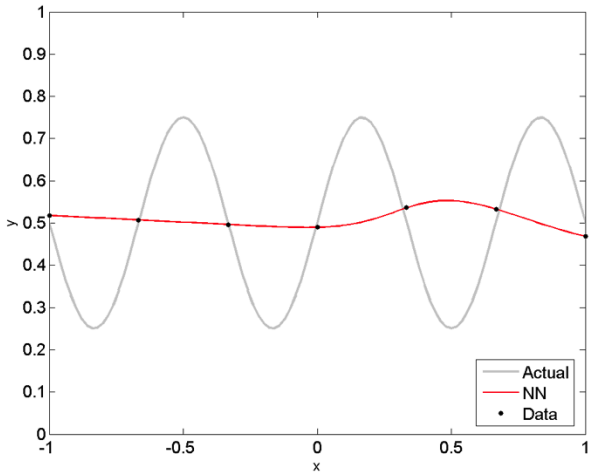

Figure 4: Network trained with 7 design points (averaged over 20 observations).

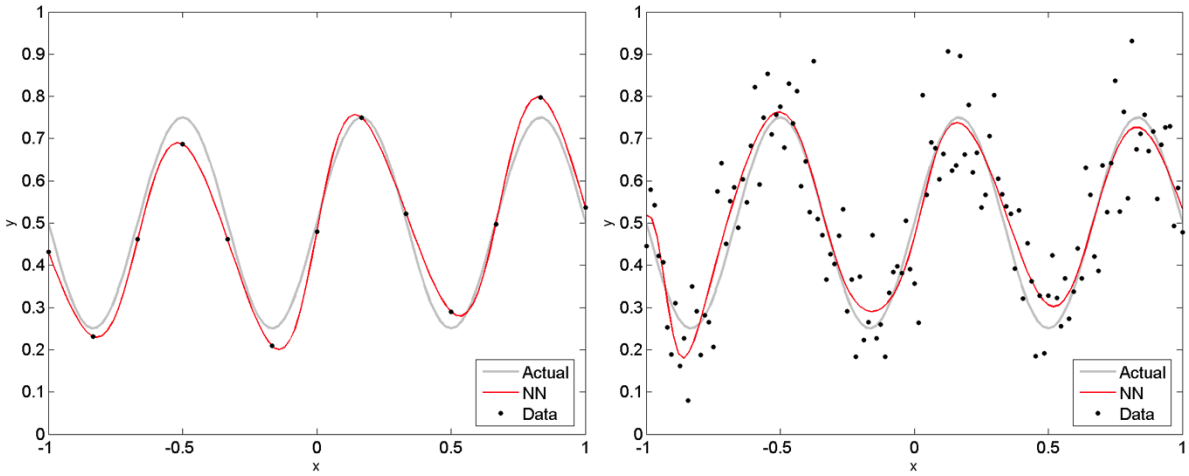

Figure 5: Networks trained on 13 design points sampled 20 times, with MSE = 0.00164 (left), and on 129 design points, each sampled once, with MSE = 0.00127.

3.3 Two Input Variable Example: Inventory System

This example is based on the reorder point, order-up-to-level inventory model presented in Law and Kelton (2000). The model simulates an inventory system where demand arrives randomly and is satisfied from stock. An inventory policy consists of the reorder point, s, and the order up to level, S. When the inventory level is at or below the reorder point, s, a replenishment order is placed for $S - I$ units, where I is the inventory level at the time of the order. These replenishments arrive after a random lead time. The output of the simulation is the total cost of the inventory policy (values of s and S), which is the sum of the ordering, carrying and shortage costs. To make the parameterization more convenient, the value of $d = S - s$ is used in the model, with $s \in \{0,\ldots,100\}$ and $d \in \{5,\ldots,100\}$. To understand the expected function to be approximated, all 9,696 possible policies (combinations of s and d) were each simulated 100 times, and the average total cost was computed (Figure 6). We will treat this as the "actual" data.

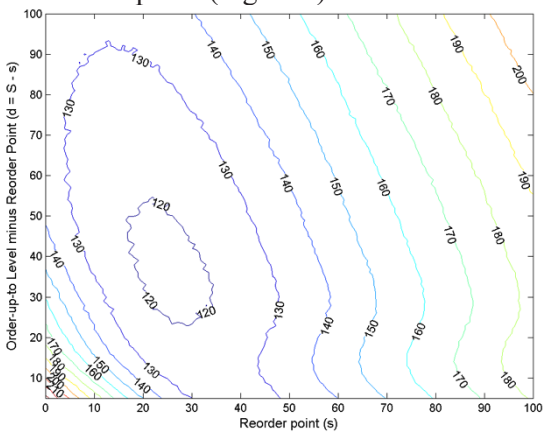

Figure 6: Contour plot of all inventory policies, output averaged over 100 replications

In the Kilmer and Smith (1993) paper discussed earlier, the 36 policies selected for their experiment included all combinations of $s \in \{0, 20, 40, 60, 80, 100\}$ and $d \in \{5, 20, 40, 60, 80, 100\}$. In this experiment, we also generated two separate training datasets. The first used the same 36 policies, each simulated 10 times and the average total cost, tc, was used as the observed output (for a total of 36 data points). The second dataset was built by randomly selecting 360 different inventory policies (combinations of s and d) and then simulating each of these policies just once to generate an observation of total cost. We then proceeded to fit a neural network to both of the datasets. Note that both data sets required the same amount of simulation effort. A contour plot of the results is shown in Figure 7.

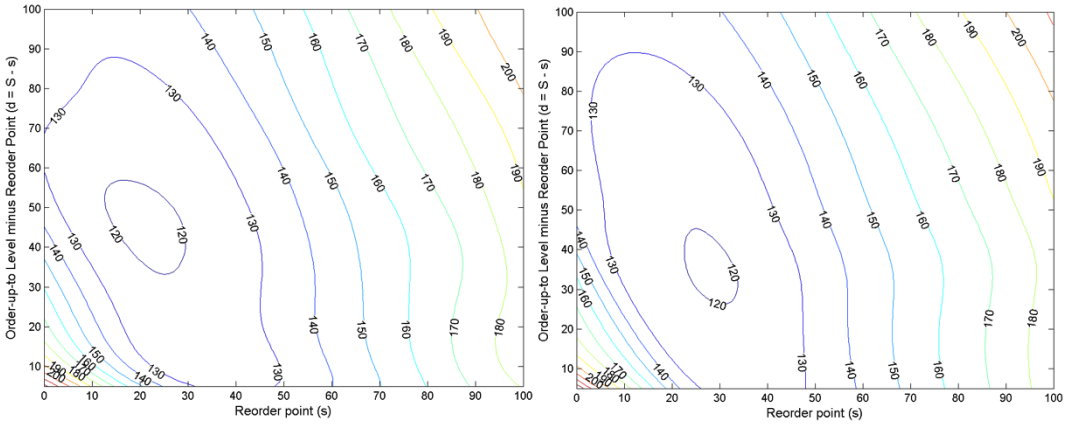

Figure 7: Contour plot of total cost for all policies, a) estimated by from network trained with 36 input points, b) estimated by network trained with 360 points

For each network, the mean squared error, mean error, and maximum error between the output of the neural network and the observed value in the training set was computed (Table 1). As expected, the MSE between network and training data was smaller for the network trained on the 36 points with the average total cost. We then computed these same error measures between the output of the each network for all 9,696 policies and the average of 100 simulations of each policy ("Validation Set").

Table 1: Measures of Error for Neural Networks for the Inventory Problem

	36 Point Network		360 Point Network	
	Training Set	"Actual" Data	Training Set	"Actual" Data
Mean Squared Error (MSE)	0.000325	2.23	7.88	1.54
Mean Error (ME)	0.000270	-0.369	-0.00100	-0.0665
Maximum Error	0.0615	9.47	12.8	7.98

When measured against the "actual" data, the 360 point network MSE (1.54) is 30% smaller than the 36 point network MSE (2.23), but its mean error (-0.0065) is 6 times closer to zero than that of the 36 point network (0.369). What is very interesting here, however, is the difference for both networks between the MSE for the training dataset and for all policies. With the 36 point network, the MSE on the actual data (2.23) is almost 10^4 times larger than the MSE observed during training, while for the 360 point network, the MSE of 1.54 is almost 80% less than the MSE observed during training. Obviously the 360 point network is not a case of overtraining.

3.4 Multiple Input Example: Design of a Production Control Strategy

The final example is a simulation model of a manufacturing system operating with the Production Authorization Card System (Buzacott and Shanthikumar 1992), a token based control system that represents kanban, base stock and local control as special instances. A control system limits the flow of materials and information in a manufacturing system in order to control work-in-process (WIP) inventory. The simulation model (MacDonald and Gunn, 2008) was used to study the impact on system performance of different control policies on a system with random customer arrivals and random processing times at each station. In this example, it is assumed that raw materials are always available, and there is always demand at the end of the line, so the final station will always be working provided the required parts are available. A control policy in this case is described by two variables per product per station: initial inventory (WIP cap) and the number of process tags at each product store. See MacDonald and Gunn (2008) for more details. The system used here has five processing stations including one assembly station (Figure 8), and therefore 10 input variables. Given the range selected for each variable, the number of possible variable value combinations (policies) is over 10 billion. The goal was to create a simulation metamodel that will predict the average amount of Work in Process Inventory (WIP) in the system for any specified policy.

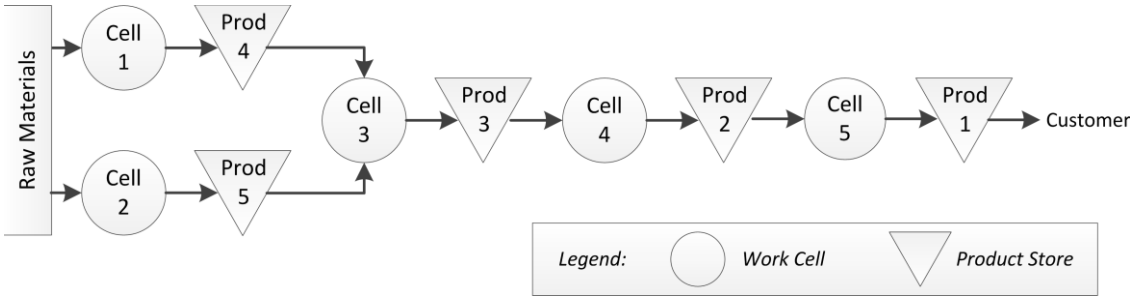

Figure 8: Schematic of the 5 Station Manufacturing Example

For this example, we created two different training datasets to use in network training. Our sampling strategy involved dividing the ranges for each input variable into subsets (either two or three, depending on the size of the range), and then using a factorial design that specified the subset for each variable in a design point; the actual value for each variable in the design point was then randomly selected from the specified subset. Each design point was then simulated 10 times, and the average of these replications was used in the training set. The second training dataset consisted of 10 randomly selected policies from each subset combination; essentially, we sampled within each small region rather than at a single point. Not every combination of these subsets can produce a feasible or desirable combination, and was therefore omitted during the sampling process (see MacDonald and Gunn 2007) resulting in 4896 feasible subset combinations. Finally, the two extreme points (all variables at a minimum, and all at a maximum) were also included. The first dataset contained 4898 points, while the second contained 48,962 points. These two datasets required roughly the same amount of simulation effort to generate the corresponding outputs. A third dataset containing another 4896 randomly selected points (independent of the initial 4898 points) was generated, and these policies were each simulated 50 times; the resulting averages are treated as "actual" values used to test the results of the two networks. The results of these tests are shown in Table 2.

Table 2: Error Measures for Production Control Strategies Example

	Network 1 (10 reps/pt)	Network 2 (No Replication)
Mean Squared Error (MSE)	0.0380	0.0241
Mean Absolute Error (MAE)	0.1019	0.0966
# Points with more than +/- 2% error	99 (2.02%)	42 (0.86%)

The network generated using the means of 10 replications (Network 1) has a MSE on the 4896 "actual" points that is about 58% larger that the MSE of the network generated using one replication on 10 times as many points (Network 2) although its MAE is only 5.5% larger. Network 1 had twice as many points that were more than +/- 2% different from the test data. Even though 4896 simulation points is a large data set, it is still small relative to the number of possible points in the input space. This example should not be regarded as a proof, but it does illustrate an instance where more accurate data does not result in a more accurate neural network.

4 CONCLUSION

The generation of a training set to be used to construct a simulation metamodel requires several important decisions: selection of the design points, specification of the warmup period and run length for each observation of the simulation output, and the number of replications per design point. Santos and Santos (2009b) showed through experimentation that the accuracy of regression metamodels is not significantly different across datasets for larger number of design points and fewer replications, provided the simulation effort is approximately the same. They further concluded datasets with fewer design points sampled with replication are preferable, given the statistical information that replication provides. However, instead of using the function form of polynomial regression, we used neural network metamodels and have shown that this conclusion may not be valid.

When training neural networks, overfitting can obviously be an issue when only a small number of (not necessarily noisy) data points are available for the training dataset. If the function is known to be smooth, then a penalty term may be used to prevent the development of an overly complex model. However, if the function is known not to be smooth, constraining the network in this way may prevent the network from fitting the data properly. Therefore, using more design points, even if the observed output is noisy, can be a better option, as shown in the examples above. If using regression, the selection of the

appropriate functional form alleviates this concern; if the selected model is a good one, only a few accurate points in the dataset are necessary.

For large complex systems, creating neural network metamodels requires a certain amount of art in terms of choice of warmup intervals, run lengths, experimental design and number of replications. However, the results of this paper show the importance of good coverage of the overall parameter space and suggest that accuracy at the design points can be sacrificed in favor of good spatial coverage.

REFERENCES

Alam, F.M., K.R. McNaught, and T.J. Ringrose. 2004. "A Comparison of Experimental Designs in the Development of a Neural Network Simulation Metamodel." *Simulation Modelling Practice and Theory* 12: 559-578.

Barton, R.R. 1998. "Simulation Metamodels." In *Proceedings of the 1998 Winter Simulation Conference*, Edited by D.J. Medeiros, E.F. Watson, J.S. Carson, and M.S. Manivannan, 167-174. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Buzacott, J.A., and J.G. Shanthikumar. 1992. "A General Approach for Coordinating Production in Multiple-cell Manufacturing Systems." *Production and Operations Management* 1(1): 34-52.

Fonseca, D.J., D.O. Navaresse, and G.P. Moynihan. 2003. "Simulation Metamodeling through Artificial Neural Networks." *Engineering Applications of Artificial Intelligence* 16: 177–183.

Geman, S., E. Bienenstock, and R. Doursat. 1992. "Neural Networks and the Bias/Variance Dilemma." *Neural Computation* 4(1): 1-58.

Hagan, M.T. and M.B. Menhaj. 1994. "Training Feedforward Networks with the Marquardt Algorithm." *IEEE Transactions on Neural Networks* 5(6): 989-993.

Hurrion, R.D., and S. Birgil. 1999. "A Comparison of Factorial and Random Experimental Design Methods for the Development of Regression and Neural Network Simulation Metamodels." *Journal of the Operational Research Society* 50: 1018-1033.

Kilmer, R.A., and A.E. Smith. 1993. "Using Artificial Neural Networks to Approximate a Discrete Event Stochastic Simulation Model." *Intelligent Engineering Systems Through Artificial Neural Networks* 3: 631-636.

Kilmer, R.A., A.E. Smith, and L.J. Shuman. 1999. "Computing Confidence Intervals for Stochastic Simulation using Neural Network Metamodels." *Computers and Industrial Engineering* 36(2): 391-407.

Kleijnen, J.P.C. 2005. "An Overview of the Design and Analysis of Simulation Experiments for Sensitivity Analysis." *European Journal of Operational Research* 164: 287-300.

Kleijnen, J.P.C. 2007. "Regression Models and Experimental Designs: A Tutorial for Simulation Analysts." In *Proceedings of the 2007 Winter Simulation Conference*, Edited by S.G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J.D. Tew, and R.R. Barton, 183-194. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kleijnen, J.P.C., S.M. Sanchez, T.W. Lucas, and T.M. Cioppa. 2005. "A User's Guide to the Brave New World of Designing Simulation Experiments." *INFORMS Journal on Computing* 17(3): 263-289.

Law, A. M., and W. D. Kelton. 2000. *Simulation Modeling & Analysis.* 3rd ed. New York: McGraw-Hill, Inc.

MacDonald, C., and E.A. Gunn. 2007. "Analyzing Manufacturing Systems using Neural Network Metamodels." In *Proceedings of the 2007 Industrial Engineering Research Conference (IERC)*, Edited by G. Bayraksan, W. Lin, Y. Son, and R. Wysk.

MacDonald, C., and E.A. Gunn. 2008. "A Simulation Based System for Analysis and Design of Production Control Systems." In *Proceedings of the 2008 Winter Simulation Conference*, Edited by S. J. Mason, R. Hill, L. Moench, and O. Rose, 1882-1890. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Saito, K., and R. Nakano. 2000. "Second-order Learning Algorithm with Squared Penalty Term." *Neural Computation* 12: 709-729.

Santos, P.R., and M.I. Santos. 2008. "Sequential Experimental Designs for Nonlinear Regression Metamodels in Simulation." *Simulation Modelling Practice and Theory* 16: 1365-1378.

Santos, P.R., and M.I. Santos. 2009a. "Construction of Stochastic Simulation Metamodels based on Unreplicated Smoothed Data." *Simulation* 85(6): 387-396.

Santos, P.R., and M.I. Santos. 2009b. "Design Experiments for the Construction of Simulation Metamodels." In *Proceedings of the 23rd European Conference on Modelling and Simulation*, Edited by J. Otamendi, A. Bargiela, J.L. Montes, and L.M.D. Pedrera, 338-344. Nottingham, UK: European Council for Modelling and Simulation.

Sjöberg, J., and L. Ljung. 1995. "Overtraining, Regularization and Searching for a Minimum, with Application to Neural Networks." *International Journal of Control* 62(6): 1391-1407.

Smith, M. 1993. *Neural Networks for Statistical Modeling*. New York: Van Nostrand Reinhold.

AUTHOR BIOGRAPHIES

CORINNE MACDONALD is an Assistant Professor in the Department of Industrial Engineering, Dalhousie University. She received a BEng in Industrial Engineering from the Technical University of Nova Scotia and a PhD in Industrial Engineering from Dalhousie. Dr. MacDonald's research and teaching interests include modeling and design of production systems, work design, simulation and metamodeling. She is a member of Canadian Operational Research Society, a Senior Member of IIE, and a registered professional engineer. Her email address is corinne.macdonald@dal.ca.

ELDON A. GUNN is a Professor of Industrial Engineering at Dalhousie University. He received a BSc from Mount Allison University, an MA from Dalhousie University, and a PhD in Industrial Engineering from the University of Toronto. Dr. Gunn's research interests involve the application of operations research models in a variety of settings, ranging from natural resources to manufacturing. He is a former President of the Canadian Operational Research Society and a Fellow of the IIE. His email address is eldon.gunn@dal.ca.