A COMPARISON OF NEURAL NETWORKS TO OLS REGRESSION IN PROCESS /QUALITY CONTROL APPLICATIONS†

Kyungdoo Nam* · Clive C. Sanford** · Maliyakal D. Jayakumar**

ABSTRACT

This study compares the performance of neural networks and ordinary least squares regression with quality-control processes. We examine the applicability of neural networks because they do not require any assumptions regarding either the functional from of the underlying process or the distribution of errors. The coefficient of determination(R^2), mean absolute deviation(MAD), and the mean squared error(MSE) metrics indicate that neural networks are a viable and can be a superior technique. We also demonstrate that an assessment of the magnitude of the neural network input layer cumulative weights can be used to determine the relative importance of predictor variables.

1. INTRODUCTION

With production and manufacturing processes characterized by different inputs, it is often beneficial to be able to identify those inputs that have the most significant influence on the quality of the process and ultimately the final prodct. Ordinary least squares(OLS) regression is a popular technique that models the functional relationship between a set of input variables and an output variable which represents the quality of the final product or an allied process parameter. OLS regression process control applications are often discussed in the literature[e.g., 5, 10], and the research usually address the modeling of the effects of input process parameters such as time and temperature on product yields.

The implementation of OLS relies on a few strong distributional assumptions about the model errors. The errors must be (1) independent, and (2) normally distributed with a zero mean and a constant variance. The predictive capability of OLS is not significantly compromised when there

* Business Administration Division University of Texas of the Permian Basin 4901 East University
** Business Computer Information Systems Department College of Business Administration University of North Texas
** Business Computer Information Systems Department College of Business Administration University of North Texas

are small departures from these assumptions. However, the model's performance deteriorates with the assumptions are violated [7, 13]. Paradoxically, such deviations can be detected only after implementation of the model. For this reason, the development of an alternative methodology that is free of any distributional assumptions merits consideration.

In recent years, neural network(NNs) have been projected in the operations research domain as an alternative to traditional statistical and optimization techniques. Neural networks are computer-based simulations of living nervous systems that hav a mathematical basis [14]. They can perceive and recognize distorted spatial and/or temporal patterns in the presence of noise with discrete or continuous data. Also, neural networks can "learn" about multidimensional probability density functions and can act as controllers in constrained environments [20]. Neural networks are therefore often used as an alternative to traditional statistical classifiers because they can make relatively weak assumptions concerning the shapes of the underlying probability density functions.

The purpose of this study is to explore the potential of using NNs in lieu of OLS regression in process control applications. We therefore present our comparative experimental results of the application of both of these techniques to a set of manufacturing processes. In general, our results favor the use of neural networks. A benefit of the NN approach is its usability with relatively small data sets and extendibility to other regression-type models.

With conventional regression analysis, the influence of an independent variable can be assessed by the observed value of the t-statistic that corresponds to its extimated coefficient. In an analogous manner, trained neural network input layer weights can help assess the predictive influence of the input variables on the output variable. Although this concept has been previously discussed [6], our experiment investigates its validity.

The following section present a brief overview of the back propagation NN algorithm. We then describe the four modeled manufacturing processes and the corresponding neural network architectures and regression model in sections 2 and 3. The experimental results are presented and interpreted in section 4 Finally, section 5 concludes this article with a discussion of the benefits and obstacles practitioners are confronted with when implementing neural networks.

2. THE BACKPROPAGATION NEURAL NETWORK

In recent years, neural networks, which are also referred to as artificial nerual systems, connectionist systems, neurocomputers, parallel distributed processing models, adaptive ystems, and self-organizing systems hav been developed as a result of the quest to overcome the limitations of artificial intelligence and algorithmic approaches in solving many difficult knowledge-processing problems [4]. The basic building block of a neural network is the neural element, or neuron, which

is utilized to construct a trainable decision-making system.

The backpropagation(BP) algorithm is central to much current work on learning in neural networks. It was invented independently several times by Bryson and Ho [2], Werbos [21], and Rumelhart, Hinton, and Williams [17, 18] as a systematic method for training multilayer neural networks.

Figure 1 shows the schematic diagram of a typical three-layer BP network which consists of two layers of weights and three layers of neurons. A BP network always has an input layer, an output layer, and at least one hidden layer. There is no theoretical limit on the number of hidden layers, and they are usually completely connected(i,e., there is an arc from each node in the lawer layer to every node in next higher layer). Typically one or two hidden layers is sufficient to handle most problems.

Let φ, η, and κ denote the sets of subscripts for the input, hidden, and output layers respectively, then

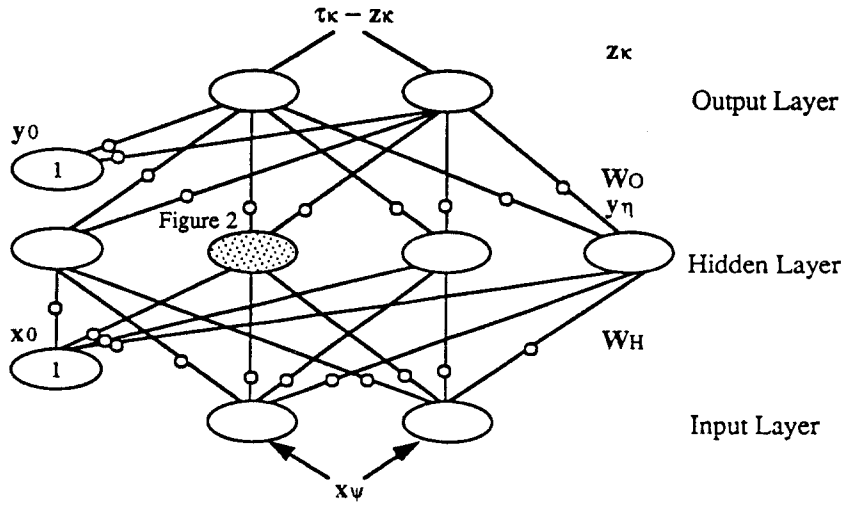
$\varphi \in \{0, 1, \cdots \phi\}$,

$\eta \in \{0, 1, \cdots \xi\}$, and

$\kappa \in \{1, 2, \cdots \pi\}$,

where ϕ, ξ, and π are the cardinalities of the input, hidden, and output layers respectively. We define $x\varphi' = (x_j \ \forall_j \in \varphi\text{-}\{0\})$, $y\eta' = (y_j \ \forall_j \ \eta\text{-}\{0\})$, and $z\kappa = (z_j \ \forall_j \in \kappa)$ as row vectors. Optional thresholding neurons in the input and output layers are defined by the unit singlets $x0 = y0 = (1)$. They control the thereshold level of the activation function(discussed later), thereby permitting a more rapid convergence of the training process. The threshold neurons are combined with $x\varphi'$ and $y\eta'$ to produce row vectors $x\varphi = x0 \cup x\varphi'$ and $y\eta = y0 \cup y\eta'$.

Figure 1. A three-layer backpropagation network configured for the training phase.



There are two phases in using the network : learning(or training) and recall. BP networks allow for supervised training with historical input, $x\varphi'$, and the desired output, $t\kappa$, sets of training vectors. The objective is to minimize the global error, E, where the target output, $t\kappa$, is compared with the output signal $z\kappa$,

$$E = \frac{1}{p}\sum |t\kappa - z\kappa|^p,$$

where p is a nonnegative real number. Most commercial BP applications minimize the least squares, and thus set p=2[e.g. 15, 16]

Network training minimizes this error by modifying weight matrices

$W_H = (w_{ij} \vee_j \in \varphi, \vee_j \in \eta)$, and

$W_O = w_{jk} \vee_j \in \eta, \vee_k \in \kappa)$,

on the two layers of arcs during training to approximate the system function. An epoch is the number of sets of input-output training vectors presented to the network between weight updates during time interval t. This time interval is often referred to as a learning cycle.

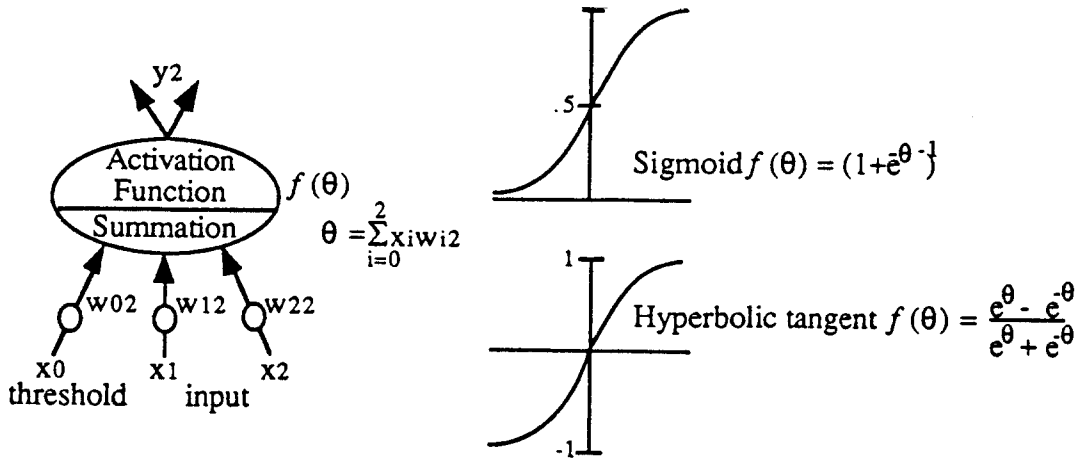
The BP algorithm updates its weights by propagating the errors backward through the network via the generalized delta rule [17, 18]. This is a gradient descent algorithm in which changes are made in the direction of the largest change in the error. The absence of arrows between the layers in Figure 1 indicates that information flows backward through the network to adjust the value of the weights between each layer. During the recall phase the signals just flow forward from the input to the output nodes.

Each weight matrix is initialized with a random number from a uniform distribution typically over R{±.3} [16], and the activations of the thresholding units are intialized to 1.0. The threshold unit values are not allowed to change during training and the input neurons are always clamped to the values of the input vector($x\varphi$). A sigmoid or hyperbolic tangent threshold activation function is applied to the cross product of $x\varphi$ and weight matrix(W_H) to generate output vector

$$y\eta = f(x\varphi W_H).$$

Backpropagation algorithms designed to adapt layered networks of neurons require differentiability along the signal paths of the network. The sigmoid and hyperbolic tangent functions are often used to accomplish this by compressing the range of the outputs between 0 and 1 or - 1 and 1 respectively. The signal processing which occurs in a standard BP hidden neuron is illustrated in Figure 2.

Figure 2. A hidden neuron with an activation function.



Inputs y_η are ultimately processed by the output neurons which generate the signal : $z_\kappa = f(y_\eta W_o)$.

2.1 Backpropagating the local error

The minimization of the global error of the network is accomplished through our knowledge of the local error at each processing element. The difference between the actual and target values($t_\kappa - z_\kappa$) produces local error signals at each neuron in the output layer. The error derivatives at the output cells(δ_κ) are generated by the dot productn of the local error signals and the derivative of the sigmoid function [$f'(y_\eta W_o) = f(y_\eta W_o)(i - f(y_\eta W_o))$], where i is the unit row vector of dimension π.

$$\delta_\kappa = f'(y_\eta W_o) \cdot (t_\kappa - z_\kappa)$$

The error derivatives are then multiplied by each activation function in the hidden layer neurons and by a learning rate coefficient β(typically between 0.01 and 1.0), which produces the gradient descent rule :

$$\Delta_o = \beta \delta_\kappa^T f(x_\varphi W_H) \tag{1}$$

where δ_κ^T is the transpose of the error derivatives. The row vector elements of λ_o are added to each corresponding column weight element of W_o after the presentation of the input-output training pairs during learning cycle t to produce adjusted weights at learning cycle $t+1$:

$$W_o(t+1) = \sum_{\forall \eta, \kappa} (W_o(t) + \Delta_o). \tag{2}$$

This iterative process adjusts the weights between the output and hidden layers. However, Δ_H, the weight adjustments between the hidden and input layers, must be generated without benefit of the target vector. To accomplish this, the error derivatives of the hidden layer($\delta\eta$) are generated by the dot product of the derivative of the hidden layer output and the cross product of the transpose of the output error derivatives($\delta\kappa^T$) and weight matrix **Wo** :

$$\delta\eta = f'(x\varphi W_H) \cdot Wo\delta\kappa^T.$$

This is used to adjust the weights feeding the hidden layer by using equations (1) and (2), modifying subscripts to indicate the correct layers.

2. 2. The momentum rate(α)

One of the problems of a gradient descent algorithmn involves the determination of the learning rate coefficient(β). Changing the weights as a linear function of the partial derivative assumes that the error surface is locally linear. At points of high curvature this linearity assumption does not apply and divergent behavior might occur near such points. It is therefore important to keep β "low" in order to avoid such behavior. However a small learning rate coefficient can lead to very slow learning, and hence Rumelhart, Hinton, and Williams [18] introduced the momentum term, α (with a range of 0.0 to 1.0), in order to resolve this dilemma. The delta weight equations (1) and (2) are modified so that a part of the previous delta weight is fed through to the current delta weight :

$$\Delta o(t+1) = \beta\delta\kappa \sum_{\forall\eta,\kappa} f(x\varphi W_H) + \alpha[\Delta o(t), \text{ and}$$

$$Wo(t+1) = \sum_{\forall\eta,\kappa} (Wo(t) + \Delta o(t+1)).$$

By reinforcing recent trends and canceling oscillatory behavior, these equations behave as a low-pass filter on the delta weight terms. This allows for a low training rate coefficient and a faster learning rate. Typically, α is set to a value of 0.9.

3. DESIGN OF THE COMPARATIVE STUDY

Our study compares the NN and OLS techniques by presenting them with input variables from four processes. A computer experiment was conducted to compare the output characteristics of the NNs to OLS.

3. 1. Modeled Processes

The experimental subjects are four processes.

P1 : Variables are from a continuous chemical process [7]. The dependent variable is the fraction of original material remaining. The independent variables are the reaction time in minutes(x_1) and the temperature in degrees Kelvin(x_2). The 38 observations in the data set were divieded randomly into 32 for the training set and 6 for the recall set.

P2 : Variables are from a continuous mechanical spray congealing process [7]. The dependent variable is the mean surface-volume particle size of the product. The independent variables are the feed rate per unit of whetted wheel periphery in gm/sec/cm(x_1), the peripheral wheel velocity in cm/sec(x_2), and the feed viscosity in poises(x_3). The data set consisted of 35 observations divided randomly into 29 observations for the training set and 6 in the recall set.

P3 : The output variable relates the production of petroleum spirit from processed crude oil to 4 independent variables : specific gravity of the crude(x_1), crude oil vapor pressure in lbs/in2 (x_2), the ASTM 10% distillation point in ^{0}F(x_3), and the petroleum fraction endpoint in 2F(x_4) [1]. The data set consisted of 32 observations divided randomly into 27 observations for the training set and 5 in the recall set.

P4 : The variables are from a degassing operation of an aqueous stream that is pumped through a nozzle into a reactor vessel in which a specified liquid level is maintained [22]. The dependent variable is the outlet gas concentration in parts per million(ppm). The independent variables are the pressure in the reactor in atmospheres(x_1), the nozzle pressure in atmospheres (x_2), the flow rate in m^3/hr (x_3), the inlet gas concentration in ppm (x_4), the liquid temperature in ^{0}C (x_5), pH in the reactor(x_6), and the liquid concentration (x_7). The data set consisted of 58 observations divided randomly into 49 observations for the training set and 9 for the recall set.

3. 2. Neural Network Architecture

We applied the backpropagation neural network algorithm because it provides a way to adjust weights in a network with many layers and has produced a number of successful applications. The backpropagation neural network was trained with Neuroshell, a software package by Ward Systems Group, Inc. [16].

The network was configured with the numbr of input(ϕ) and output (π) neurons that matched the number of input and output variables for each process. The number of hidden neurons(ξ) was determined through a preliminary analysis by presenting the training data sets over a 1 minute training period to the NNs configured with hidden neurons over the range of {2, ···, 8}. A training period of 1 minute was deemed adequate to enable this configuration to reach a stable learning

threshold [11]. The NN with the smallest number of hidden neurons that yielded an output R^2 value statistically nearest to 1 for each process was selected for use in the experiment(Table 1).

Table 1. Neural Network Architectures

Configuration	P1	P2	P3	P4
# Input neuron(ϕ)	2	3	4	7
# Hidden neurons(ξ)	6	3	8	8
# Output neurons(π)	1	1	1	1

The sigmoid activation function was employed in the hidden and output neurons. Stable learning was achieved by setting the learning rate coefficient(β) to 0.01 and the momentum term(α) to 0. 9. After training, the networks were presented with the recall data sets and ultimately produced the predicted process output variables.

3. 3. OLS Regression

For each process, the OLS linear first order regression model with m independent variables and n observations is of the form

$$y_i=\beta_0+\sum_{j=1}^{m}\beta_j x_{ji}+\varepsilon_i$$

where :

$y_i \forall i \in \{1, \cdots, n\}$ are the values of the dependent or response variables,

$\beta_j \forall j \in \{0, \cdots, m\}$ are the regression coefficients,

$x_{ji} \forall j \in \{1, \cdots, m\}\ \forall_i \in \{1, \cdots, n\}$ are the values of the independent variables, and

$\varepsilon_i \forall i \in \{1, \cdots, n\}$ are independent random error terms $N(0, \sigma^2)$

Assuming that $E(\varepsilon_i)=0$, the response function for model(3) is :

$$E(y)=\beta_0+\sum_{j=1}^{m}\beta_j x_{ji}$$

If we let $x_{io}\equiv 1$, then (4) can be written as :

$$y=X\beta+\varepsilon$$

where :

y is the n x 1 vector of observations,

β is the m x 1 vector of regression coefficients,

X is the n x m matrix of independent variables, and

ε is a vector of independent normal random variables with expectation $E(\varepsilon)=0$ and variance-covariance matrix $\sigma^2(\varepsilon)=\sigma^2 I$

Denoting the vector of estimated regression coefficients b_j $\forall j \in \{0, \cdots, m\}$ as **b** :

$(X^TX)b=X^Ty$,

then the unbiased least square estimators are :

$b=(X^TX)^{-1}X^Ty$.

Therefore, each OLS fit $\hat{y}=Xb$ was generated from the training and recall data sets [SAS, 19].

4. RESULTS AND DISCUSSION

Perfomance parameters used for the comparison of the NNs and OLS techniques are the coefficient of determination(R^2), the mean absolute deviation (MAD), and the mean square error (MSE). Due to outliers in the input data values, and thus the presence of inherent nonlinearities in Processes 2, 3 and 4, the nonparametric Wilcoxon signed rank statistic is used to test for significant differences in the model MADs.

The experimental results are listed in Table 2. In general, the performance parameters for the NNs are superior to the OLS regression. For example, the Process 1 NN R^2 recall value is 0.215 higher than for the OLS(0.991-0.776). The Wilcoxon signed rank p-value for Process 1(0.093) indicates that we can be 90% confident that the training NN MAD is significantly less than the corresponding MAD for the OLS. The Process 4 training NN R^2 value is 0.28 higher than the OLS (0.999-0.719), and the corresponding recall R^2 improved to 0.887 and 0.743 for the NN and OLs respectively. Also, MADs for the NNs are lower than the OLS when there are outliers present in the input data sets(as indicated by the relatively large MADs from the OLS fit for Processes 2, 3, and 4).

We are also interested in investigating the applicability of using the *relative* magnitudes of the trained input layer weights to help identify the most influential input variables. Table 3 lists the weights on each input arc for the four trained NNs and the observed t-statistics generated by the OLS regression procedure. The Process 1 input arc weights(10.6 and 9.8) and the magnitudes of the OLS t-statistic confirm the significance of each input variable. For Process 2 the input weights (3.2 and 4.9) and t-statistics (7.92 and -10.14) for variables x_1 and x_2 both support their significance, while NN weight (0.8) and OLS t-statistic (-0.53) confirms x_3's statistically insignificant contribution to the predictive capability of the output. When similar arguments are applied to Process 3, the NN and OLS models agree on the significance of input variables x_3 and x_4, but there is conflicting evidence with regards to the significance of variables x_1 and x_2. The Process 4

Table 2. Performance Parameters for Neural Networks and OLS Regression

	Neural Network		OLS Regression	
Process /Factor	Training	Recall	Training	Recall
P1				
R^2	0.996	0.991	0.866	0.776
MSE	$9.39*10^{-5}$	$1.17*10^{-5}$	0.003	0.007
MAD	0.008	0.908	0.032	0.044
P2				
R^2	0.935	0.975	0.906	0.974
MSE	2.978	1.014	3.740	1.068
MAD	1.212	0.908	1.388	0.978
P3				
R^2	0.958	0.967	0.960	0.969
MSE	6.042	2.572	5.690	2.412
MAD	1.726	1.240	1.582	1.474
P4				
R^2	0.999	0.877	0.719	0.743
MSE	4.562	627.400	916.390	1305.200
MAD	2.004	18.846	21.991	28.322
Wilcoxon signed Rank p-values for the recall **MADs**	P1 0.093	P2 0.589	P3 0.309	P4 0.340

NN weights range from 28.7 to 46.7, and the **OLS** fit indicates that x_1 and x_4(with t-statistics of 6.76 and 2.11 respectively) are significant at the 5% level of confidence. However, the trained NN assigns the largest input arc weight to x_4(46.7), while x_1 is assigned the lowest weight(28.7)

Because of the nonlinear distribution of Process 4's input data, we speculate that the hidden layer weights contribute more information about the significance of the input variables. The higher R^2 and lower **MSE** for the Proces 4 **NN** (versus the **OLS** model) is indicative that this approach is more appropriate for nonlinear processes.

5. SUMMARY and CONCLUSIONS

Our comparative study on process control applications demonstrates that neural networks are an alternative to **OLS** regression modeling techniques. The R^2, **MAD**, and **MSE** metrics indicate that

Table 3. Neural network(NN) input layer weights and OLS t-statistics

Proces input variables	NN input layer weights	OLS t-statistics
P1		
x1	10.6	−8.35
x2	9.8	−13.47
P2		
x1	3.2	7.92
x2	4.9	−10.14
x3	0.8	−0.53
P3		
x1	1.4	2.17
x2	2.2	1.18
x3	3.8	−4.63
x4	7.7	20.38
P4		
x1	28.7	6.76
x2	38.7	0.13
x3	44.0	0.87
x4	46.7	2.11
x5	34.9	−1.71
x6	43.8	0.12
x7	32.2	−1.54

the overall performance of neural networks is similar to and sometimes better than OLS regression. Although the experimental Processes feature process control applications, the results are generalizable to all situations where regression-type models are targeted.

When the underlying distributions of the input variables are unknown or nonlinear (e.g. Process 4), the application of neural networks should be the prefered technique. It would therefore be meaningful to compare the performance of a neural network with that of a nonlinear regression model because the neural network has a distinct advantage over a linear function in the presence of nonlinearity. A comparative study of neural networks and nonlinear regression models has the potential to diminish the reported advantages of neural networks.

Neural networks and OLS regression have similar features. The neural network input and output layer neurons correspond to OLS regression independent and dependent variables respectively, and when presented with a set of input variables, both tecniques can predict the value of an output variable. Neural networks also have several unique features. They can predict values for more

than one simultaneous dependent variable while regression analysis is appropriate for only one dependent variable. They also perform data transformations implicitly(through gradient descent), while regression models require the investigator to specify the transformations or explore the alternatives one at a time [12]. Since the **OLS** technique processes all input data simultaneously, estimation of the regression parametrs is a deterministic procedure [8].

Buke [3] suggests that neural networks may be superior to regression analysis in applications where : (1) underlying distributions are unknown or are generated by nonlinear processes, (2) regression model assumptions are violated, (3) outliers or irrelevant data exist, (4) a large number of attributes describes the inputs, (5) noise or distortion complicates analysis, and (6) estimation of statistical parameters is expensive(time consuming). Many of these characteristics can be found in production/manufacturing processes.

Another advantage offered by neural networks is the ability to employ an inversion technique that identifies optimal values for each of the control variables that would result in a selevted or ideal value of the dependent variable. That is, the neural network ca be inverted by reversing the flow of the signals and thus the backpropagation of the weight updates during training in order to find the optimal values of each of the control parameters that results in an "ideal" value of the dependent variable.

Neural networks also have some disadvantages. They require consistent data sets for training, and relatively large data sets may be required for noisy input values. However, this limitation applies to all types of preidctive models, including regression. Also, a learning procedure is required by all neural networks [3] and the internal structure of the trained neural network makes it difficult to trace the steps used to obtain the output [9].

In summary, the production processes we examined demonstrate an effective use of neural networks as an alternative to regression analysis in statistical process control applications. Inverstigation into the use of neural networks that use other paradigms such as **ART**(Adaptive Resonance Theory) along with multicollinear or heteroscedastic data that violates **OLS** assumptions are important areas for future research. In addition, the exploration of the viability of neural networks in dynamic process control settings(which do not have an a priori model) has the potential to contribute flexibility when controlling the manufacturing process.

REFERENCES

1. Atkinson, A. C., ***Plots, Transformations, and Regression***, (1985), Clarendon Press, Oxford, Uk.
2. Bryson, A. E. and Y. C. Ho, ***Applied Optimal Control***, (1969), Blaisdell, New York, NY,

USA.

3. Burke, L. I., "Introduction to artificial neural systems for pattern recognition," *Computers & Operations Research*,*18*(2), (1991), pp. 211-220.
4. Caudill, M., "Neural networks primer, part I," *AI Expert*, 2(December), (1987), pp. 46-52.
5. Crocker, D. C., *How to Use Regression Analysis in Quality Control*, 9, (1985), American Society for Quality Control, Milwaukee, WI, USA.
6. Davis, S. and B. Illingworth, "Using neural networks for statistical analysis," *Texas Instruments*, (1989), Dallas, TX, USA.
7. Draper, N. R. and H. Smith, *Applied Regression Analysis*, 2nd edition, (1981), John Wiley & Sons, New York, NY, USA, pp. 405, 519.
8. Duliba, K. A., "Contrasting neural nets with regression in predicting performance in the transportation industry," *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*, *4*, (1991), pp. 163-170.
9. Hawley, D. D., Johnson, J. D., and D. Raina, "Artificial neural system : A new tool of financial decision-making," *Financial Analysis Journal*,(1990), pp. 63-72.
10. Hotard, D. G. and J. D. Jordan, "Regression analysis is applied to improve product quality," *Industrial Engineering*,(1981), pp. 68-75.
11. Knight, K., "Connectionist Ideas and Algorithms," *Communications of the ACM*, *33*, (1990), pp. 59-74.
12. Marquez, L., Hill, T., Worthely, R., and W. Remus, "Neural network models as an alternative to regression," *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*, *4*, (1991), pp. 129-135.
13. Mendenhall, W. and T. Sincich, *A second course in business statistics : Regression analysis*, (1993), Dellen Publishing Co., San Francisco, CA, USA.
14. Nelson, M. and B. Illingworth, *A practical guide to neural nets*, (1991), Addison-Wesley Publishing Co., Reading, MA.
15. *NeuralWare*^{TM} : *Neuralworks Professional II/Plus and Neuralworks Explorer*, (1991), Neuralware Inc., Pittsburgh, PA. USA.
16. *Neuroshell*^{TM} : *Neural Network Shell Program*, (1991), Ward Systems Group Inc., Fredrick, MD, USA.
17. Rumelhart, D., Hinton, G. E., and R. J. Williams, ""Learning representations by backpropagating errors,"(*Nature*, *323*, (1986), pp. 533-536.)
18. Rumelhart, D., Hinton, G. E., and R. J. Williams, "Learning internal representations by error propagation," {*In Parallel Distributed Processing*, (1986), MIT Press, Cambridge, MA, USA.}
19. *SAS*^{TM} *User's Guide* : *Statistics*, *Version 5*, (1991), SAS Institute, Carey, NC, USA.