Expert Systems with Applications 39 (2012) 10943–10951

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Reliability estimation using a genetic algorithm-based artificial neural network: An application to a load-haul-dump machine

Snehamoy Chatterjee [a,*], Sukumar Bandopadhyay [b]

[a] Department of Mining Engineering, National Institute of Technology, Rourkela 769 008, Orissa, India
[b] Department of Mining Engineering, University of Alaska, Fairbanks, AK, USA

ARTICLE INFO

Keywords:
Systems reliability
Variable selection
Genetic algorithm
Entropy
Learning parameters

ABSTRACT

In this study, a neural network-based model for forecasting reliability was developed. A genetic algorithm was applied for selecting neural network parameters like learning rate (η) and momentum (μ). The input variables of the neural network model were selected by maximizing the mean entropy value. The developed model was validated by applying two benchmark data sets. A comparative study reveals that the proposed method performs better than existing methods on benchmark data sets. A case study was conducted on a load-haul-dump (LHD) machine operated at a coal mine in Alaska, USA. Past time-to-failure data for the LHD machine were collected, and cumulative time-to-failure was calculated for reliability modeling. The results demonstrate that the developed model performs well with high accuracy ($R^2 = 0.94$) in the failure prediction of a LHD machine.

1. Introduction

Estimation of reliability plays an important role in performance assessment of any system. Reliability predictions are important for various purposes, like production planning, maintenance planning, reliability assessment, fault detection in manufacturing processes, and risk and liability evaluation (Zheng, 2009; El-Sebakhy, 2009; Yeh, Lin, & Chung, 2010; Hu, Si, & Yang, 2010; Fonseca & Knapp, 2000; Lolas & Olatunbosun, 2008). Modeling of machine reliability has been one of the most important issues in mining and other industries. Knowledge of reliability beforehand allows a more accurate forecast of appropriate preventive and corrective maintenance.

The most widely applied system reliability models are based on lifetime distribution models, fault tree analysis, and Markov models. Each of these approaches has advantages and limitations. Traditional reliability models are limited in the variation of individual systems under dynamic operating conditions (Lu, Lu, & Kloarik, 2001). Moreover, most existing analytical system-reliability model processes depend on certain a priori assumptions. These assumptions are difficult to validate for real-life problems; hence, the predictive performance of different models varies. Apart from a priori assumptions, system reliability is influenced by a number of factors like systems complexity, systems development environment, and systems development methodology. These factors make system reliability models more complex and nonlinear in nature. In such situations, traditional reliability models, which assume systems are independent and linear in nature, fail to provide satisfactory results for reliability estimation.

System reliability, in general, changes with time. Some researchers have considered these changes as a time series process. The autoregressive moving average (ARMA) model has been one of the most popular approaches in time series prediction (Box & Jenkins, 1976). Ho and Xie (1998) developed an ARMA model to analyze the failures of repairable systems. Ho, Xie, and Goh (2002) applied the ARMA model with three other models to predict failures of repairable systems.

Recently, artificial neural networks (ANNs) have received growing attention in time series forecasting. The ANN, a widely used data-driven technique, is presently preferred over traditional methods of time series forecasting (Lolas & Olatunbosun, 2008; Xu, Xie, & Tang, 2003; Yi-Hui, 2007). The beauty of neural networks (NNs) is their flexibility in nature and their ability to capture complex nonlinear relationships between input and output patterns through appropriate learning. However, the application of ANNs to reliability forecasting is limited. Liu, Kuo, and Sastri (1995) who used backpropagation NN models to estimate parameters of reliability distribution, found that this method significantly improved the accuracy of reliability forecasting when compared with traditional methods. Amjady and Ehsan (1999) developed a NN-based expert system for evaluating the reliability of power systems. A study by Su, Tong, and Leou (1997) who proposed a NN-based ARMA model to predict engine reliability, showed that the proposed model performed better than the ARMA model. Xu et al. (2003) applied feed-forward multilayer perceptron (MLP) neural networks and radial basis function (RBF) neural networks to predict engine reliability. Experimental results demonstrated

* Corresponding author. Tel.: +91 661 2462610.
E-mail address: snehamoy@gmail.com (S. Chatterjee).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.03.030

that the proposed model predicted performance better than the ARMA model. The results of a study by Chang, Lin, and Pai (2004), who developed a hybrid neural-fuzzy system to predict engine system reliability, demonstrate that the model achieved more-accurate prediction than the ARMA and generalized regression NN model.

The main limitation of any data-driven model including the NN model is the problem of properly fitting the underlying distribution of the data (Bishop, 1998). Two different types of fitting problems are cited in the NN literature: under-fitting and over-fitting. Fitting problems normally are encountered because of poor selection of neural learning parameters, which include learning rate, momentum, hidden node size, and number of hidden layers. While under-fitting leads to poor network performance with training data, over-fitting leads to poor generalization of the model, which means that the model may work well with training data, but perform poorly in predicting unseen data. Hence, the main challenge for any NN model user is to suitably choose learning parameters that help restrain under- or over-fitting, as both are equally disastrous. The trial-and-error approach to selection of learning parameters is most widely used in NN problems, but this approach would not guarantee the optimum model solution for this study because it leads to development of poor network output for unseen data. In addition, computations with the trial-and-error method take more time. Our goal was to generate a systematic methodology capable of automatically selecting learning parameters for the network model in minimum time.

We used a genetic algorithm (GA) for the selection of NN parameters in reliability estimation of a load-haul-dump (LHD) machine. The capability of the GA was examined to find network learning parameters—learning rate and momentum (η and μ)—that produce the global best solution for LHD reliability analysis. We used pre-established network architecture with full connection. The same model was run with different numbers of hidden nodes to discover the most suited number of hidden node sizes. The number of input variables for the NN was selected by maximizing the mean entropy value.

This paper is organized as follows: Section 2 presents a brief overview of the steps involved in the methodology. Validation of the developed model is presented in Section 3. Section 4 presents the case study for the proposed method. Conclusions are given in Section 5.

2. Methodology

2.1. Reliability estimation by time series analysis

In time series reliability analysis, a data-driven model is trained to learn the relationship between past historical reliability indices and corresponding targets. The developed model is then used to predict future reliability values. In forecasting univariate time-series reliability, inputs used by any model are the past lagged observations of the time series, while the outputs are the future values. Each set of input patterns is composed of any moving fixed-length window within the time series.

Let $Y = [y_1, y_2, \ldots, y_t]$ is the time-series reliability data of any machine. If p is number of input variables for the time series model, then the data set for developing the reliability model can be extracted from Y in the following form:

$$T_i = \{y_{(1+i)}, y_{(2+i)}, \ldots, y_{(k+i)}, \ldots, y_{(p+i)}, y_{(p+i+1)}\},\quad i \in \{0, 1, 2, \ldots, t-p-1\},\ \ k < p \tag{1}$$

The general time series reliability-forecasting model can be represented as

$$y_{(p+i+1)} = f(y_{(1+i)}, y_{(2+i)}, \ldots, y_{(p+i)}),\quad i \in \{0, 1, 2, \ldots, t-p-1\} \tag{2}$$

where $\{y_{(1+i)}, y_{(2+i)}, \ldots, y_{(p+i)}\}$ is a vector of lagged variables, $y_{(p+i+1)}$ is the observation at time $t = (p + i + 1)$, $i \in (0, 1, \ldots, t-p-1)$, and p represents the number of past observations related to the future value. The goal of this study is to approximate the function f of Eq. (2) using the NN model. The NN approach explores the appropriate internal representation of the time-series reliability data.

In time-series reliability analysis, the NN model is trained to learn the relationship between historical reliability data and corresponding data. Future failures can then be predicted using the developed model. Training patterns can be extracted from the historical time-series reliability data Y using Eq. (1). From Eq. (2), observe that total $(t-p)$ numbers of data sets can be extracted from Y, where p denotes the number of lagged variables, and t denotes total number of reliability data observed in time series Y.

2.2. Optimum lag data selection

The main task before modeling reliability data using the time series method is to fix some criteria for selection of lag data p. The size of p should depend on small-scale structures as well as the large-scale structure present in the time series data. If the size of p chosen is very small with respect to the actual features, the statistics will differ strongly for each vector of lagged variables. Therefore, the size of p should be chosen as small as possible to improve small-scale structures, but it should be chosen as large as necessary to represent the actual features occurring in the time series data.

Entropy is a popularly applied tool for template selection in the image analysis field (MacKay, 2003). An entropy map provides insight into the features of an image and is used to reveal the template size. In this study, the concept of entropy was used to find the optimum lag data size. However, instead of a two-dimensional image, a one-dimensional time series was considered. Entropy is a statistical measure of randomness that can be used to characterize the texture of an image. The mean entropy of data set T_i with the dimensions of $(p + 1)$ can be computed as (Honarkhah & Caers, 2010)

$$H = \frac{1}{K}\sum_{i=1}^{K} q_i \log(q_i) \tag{3}$$

where K equals the number of possible outcomes of the random variable, and q_i represents the probability mass function. The high entropy represents more randomness.

Generally, as the size of lag data grows, the entropy of the data set T_i image increases. However, after a certain lag data size, the entropy value reaches a steady state. Therefore, we selected the lag data value after no further improvement of the entropy value was observed.

2.3. Overview of the neural network algorithm

After selecting the lag data value, time series data were extracted from the reliability data using Eq. (1). The reliability forecasting model was developed using an ANN algorithm and a backpropagation NN algorithm. Fig. 1 shows the three-layer ANN used in this study. Description of NN fundamentals, well documented in the literature (Bishop, 1998; Hagan, Demuth, & Beale, 1995; Haykins, 1999), is beyond the scope of this paper. The input nodes (variables) for the NN model are $\{y_{(1+i)}, y_{(2+i)}, \ldots, y_{(p+i)}\}$ and the output node (variable) is $y_{(p+i+1)}$. The hidden layer, which was not a predefined number, was fixed during the training. The adaptive gradient descent with a momentum learning algorithm was used for error minimization. This algorithm starts with an untrained network, presents a training pattern to the input layer, passes signals through the net, and determines output at the

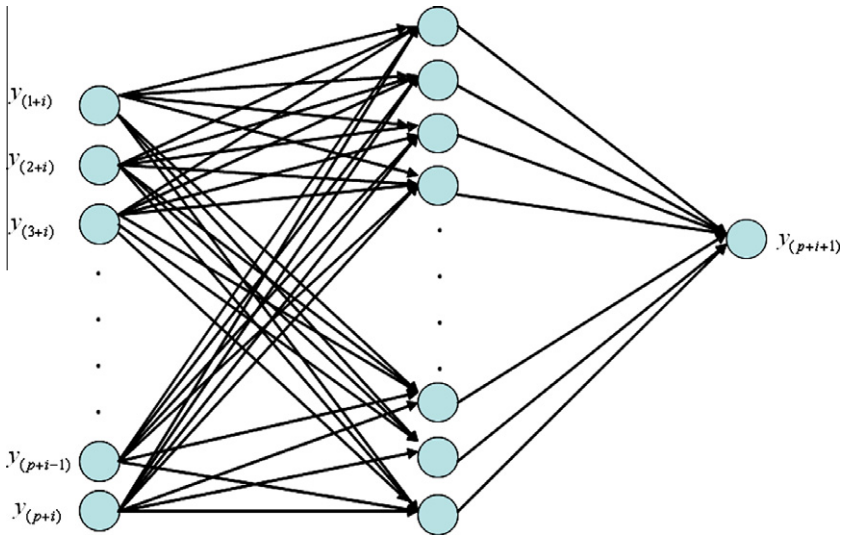

Fig. 1. Three-layered neural network architecture.

output layer. These outputs are compared with the target values such that any observed difference between them corresponds to an error. The error function, which is a scalar function of the weights, is minimized when the network outputs match the desired output. Thus, the weights are adjusted to reduce this measure of error. The error on a pattern to be the sum over output units of the squared difference between the desired and actual output are given by

$$E(w) \equiv \frac{1}{2}\sum_{i=1}^{c}|(y_{(p+i+1)} - \hat{y}_{(p+i+1)})|^2 \tag{4}$$

where $y_{(p+i+1)}$ and $\hat{y}_{(p+i+1)}$ are the target, and the network output vectors of length, c, and w represent the weights in the network. The weights are initialized with random values and then changed in a direction that will reduce the error:

$$\Delta w(m) = -\eta\frac{\partial E(m)}{\partial w} + \mu\Delta w(m-1) \tag{5}$$

where η is the learning rate and μ is momentum.

The iterative algorithm requires taking a weight vector at iteration m and updating it as

$$w(m+1) = w(m) + \Delta w(m) \tag{6}$$

The weight update process is performed every iteration until the error function reaches its threshold value. It is clear from the above discussion that the two parameters—learning rate (η) and momentum (μ)—are important criteria for proper learning of the network. Generally, parameter values are selected by trial and error, but this method may produce a bad network model. The GA was used for selection of parameters for ANN learning.

2.4. Genetic algorithm for neural network training

The genetic algorithm (GA) is a heuristic search algorithm used for searching best learning parameters (Holland, 1975); it is also the most popular form of evolutionary algorithm used in the diversified field of optimization problems (Dengiz, Atiparmak, & Smith, 1997; Li, Balazs, & Parks, 2007; Wei, Ma, Hong-Jun, Wang, & Yun-Tao, 2003). The algorithm first initializes with a population of solutions known as chromosomes, and transforms itself by three genetic operators—selection, crossover, and mutation—to get a better solution for the problem after each generation. The fitness function is used for evaluating the probability of acceptance of individual chromosomes in the next generation. Based on individual fitness values, some chromosomes are selected by elitism and some are selected for the crossover operation. Chromosomes selected for the crossover operation are known as the parent solution; after crossover, a set of parents produces two children solutions. Mutation generally changes the binary bit value, either from 0 to 1 or from 1 to 0, randomly, based on a user-selected mutation rate. The mutation operation helps the algorithm to escape the solution from local minima. After performing all three genetic operations, fitness values are calculated and the worst solutions are eliminated from the population, which helps to maintain the population size constant. This one cycle of operation is known as a generation. The obtained population of chromosomes after one generation will be the starting solution for the next generation. This operation continues until it reaches the predefined generation number or threshold criteria.

The fundamental reason for using a GA here is to determine the global minimum zone (also termed *basin*). The gradient descent algorithm, which is a local optimization method, tries to find the optimum solution within that basin. The GA implemented in this study has seven different steps.

2.4.1. Representation

The chromosomes represented both parameters—learning rate (η) and momentum (μ)—with a binary coded number. An 8-bit binary code number was used for representing 1 parameter. Therefore, each chromosome consisted of 16 bits, 8 for each learning parameter. For example, 10011011 01101011 is a single chromosome in which the first 8 digits represent the learning rate and the second 8 digits represent the momentum value for NN learning.

2.4.2. Initialization

The chromosome was uniformly initialized. The number of chromosomes in the population, that is, the population size, totaled 50.

2.4.3. Selection

A probabilistic selection is used based on the individual's fitness such that the better individuals have higher chances of being selected. In this study, the normalized geometric ranking scheme $p_i = q'(1-q)^r$ was applied, where p_i represents the probability of the ith individual being selected, q is the probability of selecting the best individual, r is the rank of the individual, and p is the population size.

2.4.4. Crossover

The crossover operation is performed in each generation to generate a better solution from available solutions. This operation is performed by interchanging genetic material of chromosomes in order to create individuals that can benefit from their parents' fitness. In this study, a uniform crossover with probability rate 0.1 was used.

2.4.5. Mutation

Mutation is the genetic operator responsible for maintaining diversity in the population. Mutation operates by randomly "flipping" bits of the chromosome, based on some probability. The mutation probability used in this study was $1/p$, where p is the length of each of the two parts of the chromosomes.

2.4.6. Random immigrant

A random immigrant introduces diverse solutions in the population and minimizes the risk of premature convergence (Congdon, 1995). Individuals having low fitness value are deleted from the population and replaced by the same number of recently initialized random individuals. In this study, the number of individuals deleted and the number of new individuals initialized in each generation totaled 5.

2.4.7. Fitness

The normalized mean squared error (NRMSE) of the validation data set was used as the fitness function for genetic learning. The k-fold cross-validation method was adopted to improve the generalization capabilities of the developed NN model. The fitness function can be represented as the average of the NRMSE of the k-fold cross-validation data:

$$\text{Fitness function} = \frac{1}{k}\sum_{i=1}^{k}\sqrt{\frac{\sum_{j=1}^{n}(y_{ij}-\hat{y}_{ij})^2}{\sum_{j=1}^{n}y_{ij}^2}} \tag{7}$$

where y_{ij} and $\hat{y}_{ij}$ are actual and predicted reliability values, respectively, and n is the number of validation data at the kth fold.

The fitness values of all these chromosomes were evaluated using the fitness function. Some of the chromosomes were selected by elitism. The probabilistic-selection criterion was applied for selecting chromosomes for the crossover and mutation operation. Some poorly fitted chromosomes were eliminated from the chromosome solution to maintain the population size constant. The initial population size was 50; after each generation, poor solutions were eliminated to maintain a population size of 20. This genetic operation was performed until it reached the maximum generation value of 50. After reaching maximum generation, the model provided a set of final solutions (50). The chromosome corresponding to the minimum error value is the best solution for the model. That chromosome has the best learning parameters (learning rate and momentum) for the ANN model. Note that hidden layer nodes are not considered a learning parameter in this work. The single hidden layer network was used.

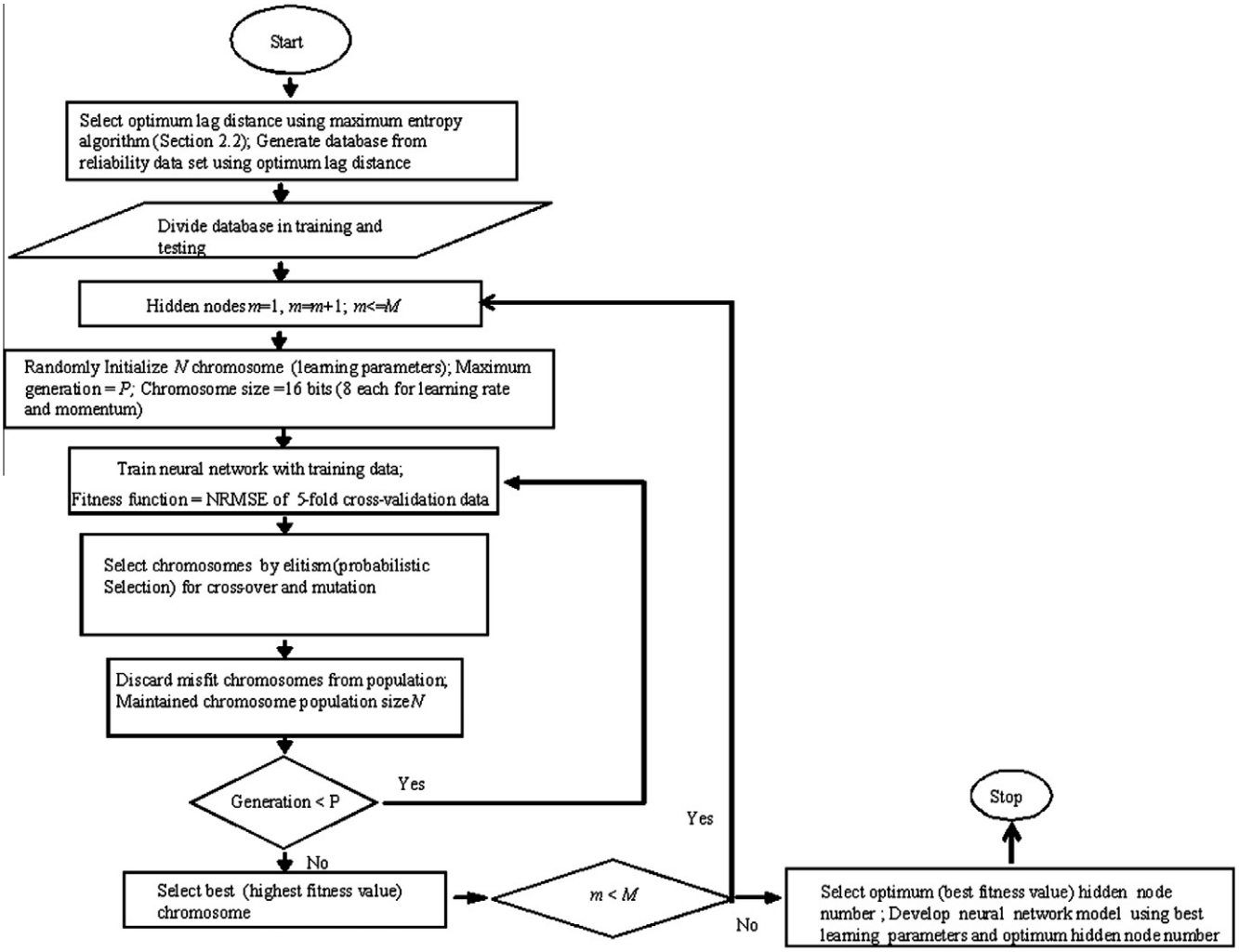

Fig. 2. Proposed algorithm applied in this study.

The next task involves choosing the hidden node size. The same GA coupled with the ANN model was used with m of different hidden node sizes. The value of m varies in this problem from 1 to 25. The error value of the validation data set was then calculated with a different hidden node size. The minimum error values corresponding to the hidden node size were selected as the best suited hidden node size, and the corresponding learning parameters were the optimum parameters for the ANN model. The flowchart in Fig. 2 represents the proposed method applied.

3. Validation of the developed model

To validate the proposed genetic algorithm-based NN training model, two data sets were used: (1) turbo-charged diesel engine failure data (Pai, 2006; Xu et al., 2003), and (2) miles-to-failure of car engine data (Xu et al., 2003), which were used to investigate the performance of the proposed model in forecasting the reliability of cars. All runs were performed on a 3.2 GHz Intel(R) Core (TM) i5 PC with 3 GB of RAM. All ANN and GA techniques were performed at R-platform. An R-code was developed by combining the GENLAG (Willighagen, 2005) and AMORE (Limas, 2006) libraries to perform the task.

3.1. Reliability forecast of turbochargers in diesel engines

The turbocharger is a vital component in the diesel engine. Reliability information on turbochargers is summarized in the Appendix.

The ANN model to estimate the reliability of turbochargers consists of input, hidden, and output nodes. The vector of lagged reliability values was used as the input for the model; the corresponding reliability value was considered the output of the model. The time-series reliability data shown in Eq. (2) are extracted from the reliability data after selecting the lag value p. The p value is selected by calculating the mean entropy values as discussed in Section 2.2. The mean entropy values of data set T_i with the dimensions of $(p+1)$, $p \in \{2, 3, \ldots, 10\}$ are presented in Fig. 3. Observe from the figure that the optimum dimension of data set T_i is 4. Therefore, the lag value p is 3 for this problem.

Thirty-seven time series data were generated from the reliability data of the Appendix after selecting the p value 3. Out of 37 data, 32 were used as training data for NN model development and 5 were selected for testing the model.

All input and output variables are normalized between 0 and 1 before NN training. The tan-hyperbolic activation function was used in the hidden layer; the pure linear activation function was used in the output layer. An adaptive gradient descent with momentum learning algorithm was used for NN model development.

The parameters of ANN learning were fixed by GA learning. A 16-bit binary code was used to represent the learning parameters (η and μ) of the NN. A population of 50 such binary codes, known as chromosomes, was randomly generated to represent the initial solution. Neural network training was performed with that population. After completion of training, populations were ranked based on their fitness values. A 5-fold cross validation is performed to calculate the NRMSE value of the validation data. The optimum values of η and μ after GA learning were 0.026 and 0.76, respectively.

For selecting the hidden node number, all parameters except the hidden node number are kept constant. The NN models were trained using a GA by varying hidden node sizes. Fig. 4 presents the NRMSEs of cross-validation data for a different hidden node number. The figure shows that, for this problem, the hidden node number is 17.

After selecting the learning parameters (η and μ) and hidden node size, the NN model was developed. The actual values and the predicted values of the training data are presented in Fig. 5, which shows that the predicted values agree well with actual values.

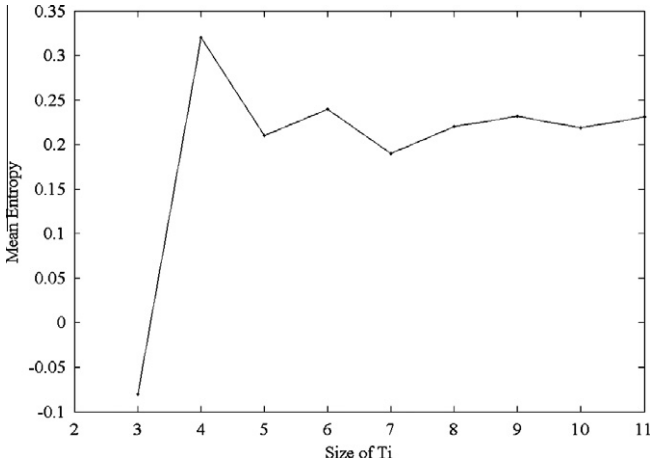

Fig. 3. Mean entropy value of different T_i size.

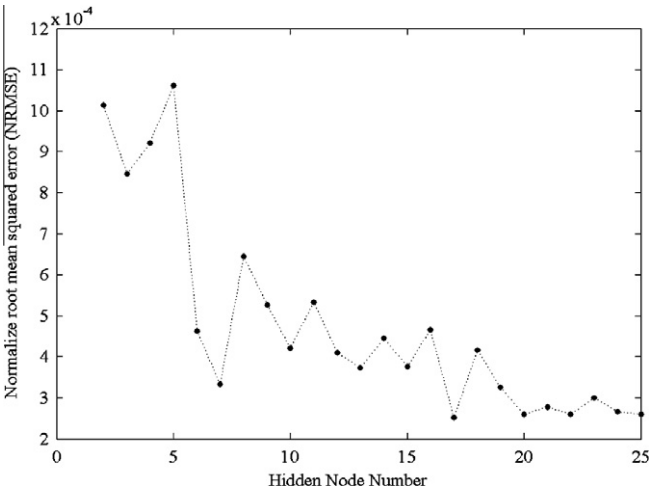

Fig. 4. Error values of the ANN model by changing hidden node number.

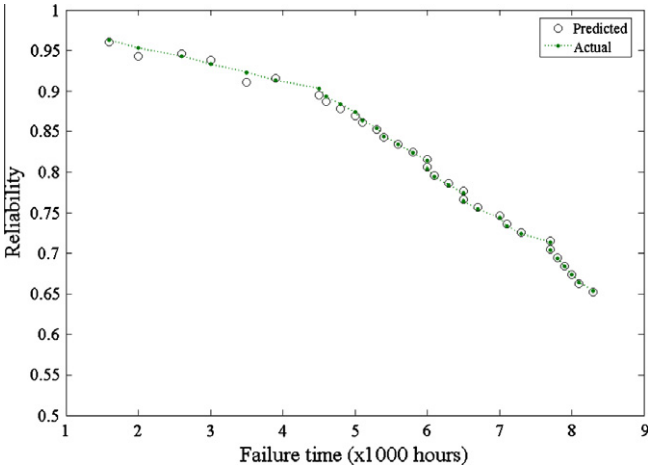

Fig. 5. Actual and predicted reliability values of the training data from the turbocharger data.

Table 1
Forecasting results of turbochargers reliability using different neural network architectures, AR model, SVMG model and our proposed method.

Number	Reliability (actual)	MLP (logistic)	MLP (Gaussian activation)	RBF (Gaussian activation)	AR	SVMG	Our proposed method
36	0.6444	0.6589	0.6515	0.6466	0.6460	0.6447	0.6441
37	0.6345	0.6532	0.6446	0.6369	0.6392	0.6346	0.6345
38	0.6245	0.6479	0.6383	0.6270	0.6338	0.6247	0.6246
39	0.6145	0.643	0.6328	0.6170	0.6293	0.6147	0.6145
40	0.6046	0.6384	0.6278	0.6072	0.6256	0.6047	0.6044
NRMSE		0.039618	0.0249653	0.003912	0.019887	0.000312	0.000252

A test data set was used to compare the proposed method with already developed methods. Table 1 presents the actual and predicted reliability values of the test data set. The reliability estimated values of MLP (logistic), MLP (Gaussian), RBF (Gaussian), and AR (Auto regressive) were obtained from Xu et al. (2003). The reliability estimated values of the SVMG (support vector machine with genetic algorithm) were obtained from Pai (2006). Observe from the table that the proposed method performed better than all presented methods.

The presented example shows that the proposed approach performed better than other algorithms in the reliability analysis of the machine. However, like other existing methods, the success of the proposed method depends on the value of p. The value of p is selected using a mean entropy value, which is subjective in nature. To see whether the selected p value is optimum, we calculated the NRMSE of the test data by varying the p value. Fig. 6 presents the error values of test data set for different p values. The figure shows that the error value is lowest when the p value is 3. Note from the mean entropy calculation that the p value selected also is 3.

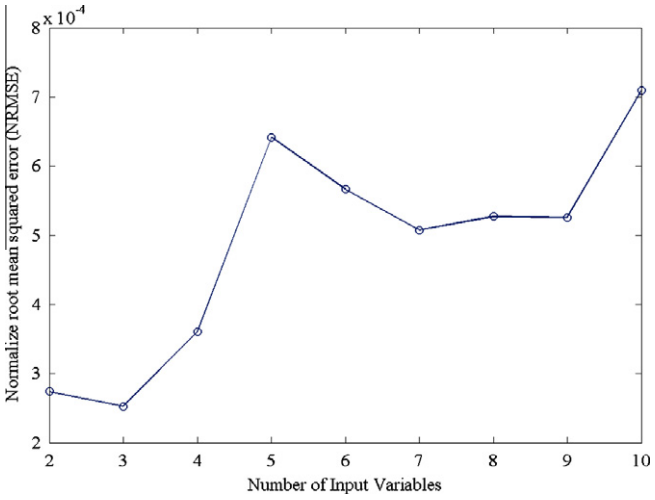

Fig. 6. Error values of test data set by varying the p value.

3.2. Forecast of the miles-to-failure of car engine data

Miles-to-failure car engine data from Xu et al. (2003) were collected to investigate the performance of the proposed model in forecasting the reliability of a car. The miles-to-failure data for 100 units of a specific brand of car engine were collected. The lagged value p selected in this data set was 4. The last 10 data from 91 to 100 were used as test data. The learning parameters, η and μ, were 0.02 and 0.787, respectively. The hidden node number was 19. Table 2 summarizes the predictive performances of the MLP, RBF, AR, and proposed model after training the network using the proposed method with optimum parameters and the hidden node number. Observe from the table that the proposed method produces the lowest NRMSE value compared with other methods.

4. Case study

The case study for the proposed method involved failure prediction of a load-haul-dump (LHD) machine in the mining industry, an industry in which LHD machines play an important role. The cutting machine cuts coal or mineral and the LHD machine loads the coal or mineral, hauls the load for some distance, and dumps it in a strategic location for further transportation. If the LHD machine fails to operate for some reason, the mining operation stops. Therefore, mine management gives full attention to avoiding LHD machine failure. If management can predict the next likely occurrence of failure, suitable maintenance can be planned and performed to avoid such failure.

Historical time-to-failure data of a LHD machine from an Alaska mine were collected—40 failure data altogether. Cumulative time-to-failure of the LHD machine was calculated. Time-to-failure data along with cumulative time-to-failure data are presented in Table 3. Out of 40 cumulative failure data, the first 34 data were used for training the algorithm and the remaining 6 were used as test data.

The method used for ANN model development is discussed in Section 2. The p value was obtained by observing the mean entropy values, which showed that the lag value p was 4 (Fig. 7). The training and testing patterns for ANN modeling were extracted from Table 3 data. All input and output variables were normalized

Table 2
Forecasting results of car engine reliability using different neural network architectures, AR model, and our proposed method.

Number	Miles-to-failure (actual) (×1000 miles)	MLP (logistic) (×1000 miles)	MLP (Gaussian) (×1000 miles)	RBF (Gaussian) (×1000 miles)	AR (×1000 miles)	Our proposed method (×1000 miles)
91	36.8571	37.0259	36.9917	37.1096	37.3546	36.98993
92	37.0476	37.8178	37.4988	37.7532	36.3213	37.54473
93	37.9048	37.8254	37.7549	38.0020	37.0603	37.58313
94	38.1905	38.5449	38.4238	37.9306	38.3013	38.11773
95	39.5238	38.8205	39.4327	38.2064	38.7083	38.82437
96	35.4286	36.5156	36.1145	36.9375	39.3056	36.03413
97	36.0000	36.4749	36.1145	36.3272	34.1770	36.59814
98	37.7143	36.8491	36.9533	36.8791	36.3164	37.13175
99	38.0952	37.9428	37.2913	37.2870	38.3696	37.87646
100	38.5714	38.7185	38.3821	38.1781	40.1801	38.36903
NRMSE		0.0156	0.0122	0.0211	0.0422	0.01195

Table 3
Failure data of load-haul-dump (LHD) machine.

Failure order number (i)	Time-to-failure (h_i)	Cumulative time-to-failure (H_i)
1	10.2	10.2
2	9.6	19.8
3	7.4	27.2
4	17.9	45.1
5	1	46.1
6	3.8	49.9
7	26.6	76.5
8	1.9	78.4
9	13.2	91.6
10	1.2	92.8
11	59.9	152.7
12	8.5	161.2
13	1	162.2
14	19.5	181.7
15	26.8	208.5
16	11	219.5
17	6.7	226.2
18	15.3	241.5
19	14.3	255.8
20	47	302.8
21	19.9	322.7
22	14.4	337.1
23	5.7	342.8
24	6.8	349.6
25	2.4	352
26	7	359
27	7.8	366.8
28	39.7	406.5
29	11.7	418.2
30	2	420.2
31	2.2	422.4
32	5.8	428.2
33	10.5	438.7
34	40.9	479.6
35	4.7	484.3
36	40.7	525
37	3.9	528.9
38	22.3	551.2
39	12.2	563.4
40	14.9	578.3

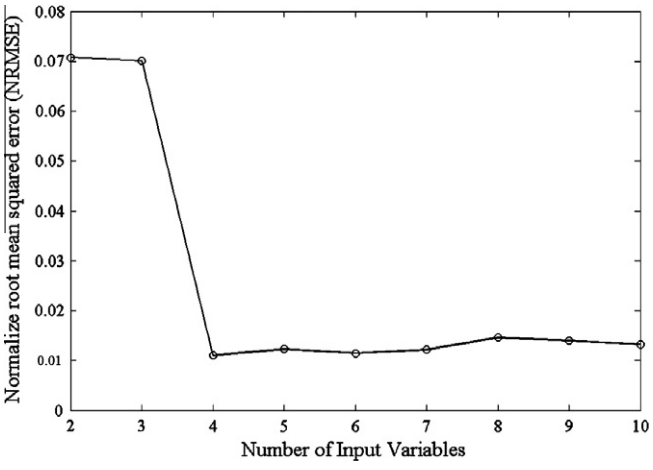

Fig. 7. Error values of forecasted cumulative LHD failure time by varying the p value.

between 0 and 1 before the NN training. The tan-hyperbolic activation function was used in the hidden layer, and the pure linear activation function was used in the output layer. The learning parameters (η and μ) were selected using the algorithm discussed in Section 2.4. The values of η and μ after genetic algorithm learning were 0.0326 and 0.598, respectively. The hidden node number was calculated and resulted in 12 (Fig. 8).

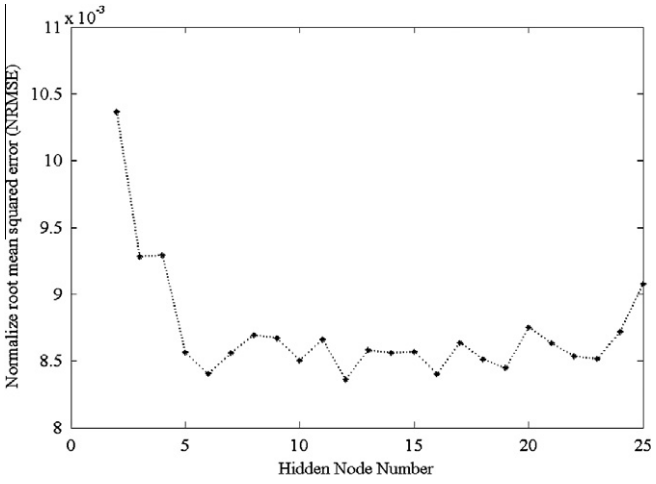

Fig. 8. Error values of 5-fold cross-validation cumulative LHD failure time data by varying hidden node number.

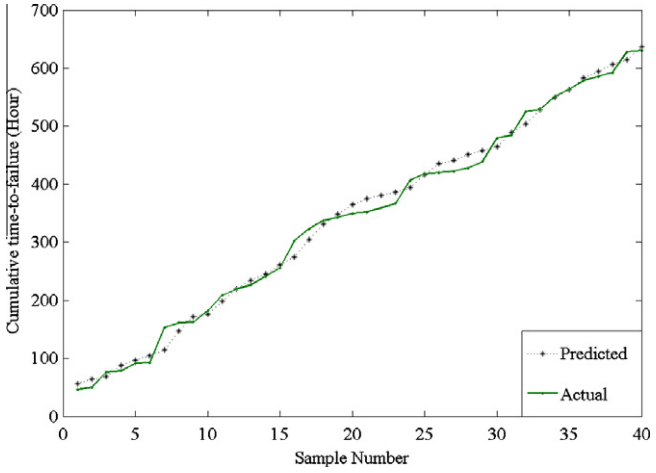

Fig. 9. Actual and predicted cumulative time-to-failure training data from the case study LHD machine.

Table 4
Training and testing data error statistics of our proposed model for reliability of LHD.

	Training	Testing
Mean error	−2.1639	0.3806
Mean absolute error	11.8141	4.0866
Mean squared error	206.15	31.3776
Error variance	206.63	37.4793
R^2	0.993	0.944

After selecting the learning parameters and hidden node number, the NN model was developed for LHD cumulative-time-to-failure data. The actual values and the predicted values of the training data are presented in Fig. 9, which shows that the predicted values closely match the actual values.

After the reliability model was developed, the generalization capability and the performance of the model were examined using the test data set. The error statistics of the actual values and the model-predicted values for the training and test data set are presented in Table 4. The mean squared error values for the training and test data are 206.15 and 31.38, respectively. The R^2 values

Table 5
Paired sample *t*-test between actual vs. predicted values of training and test data set of proposed model.

	Paired samples *t*-test							
	Mean	Std. deviation	Std. error of mean	95% Confidence of the difference		*t*-Statistic value (*t*)	Degree of freedom (*df*)	Significant level (2-tailed)
				Lower	Upper			
Training data	−2.164	14.374	2.273	−76.6285	80.9564	0.0547	78	0.9565
Testing data	0.381	6.122	2.499	−30.3096	31.0708	0.0276	10	0.9785

Table 6
Forecasting results of LHD failure using different neural network architectures, AR model, and our proposed method.

Number	Cumulative time-to-failure (actual) (h)	MLP (logistic) (h)	MLP (Gaussian) (h)	RBF (Gaussian) (h)	AR (h)	Our proposed method (h)
45	639.1	647.81	647.96	647.19	648.64	640.01
46	649.3	652.02	651.42	655.84	652.03	650.77
47	655.8	668.10	668.03	665.35	669.02	664.33
48	681.7	672.73	674.56	672.63	674.84	671.63
49	691.4	696.45	695.52	694.21	695.99	688.07
50	700.6	708.22	707.95	706.74	708.75	700.81
NRMSE		0.01216	0.01147	0.01102	0.01231	0.00836

for the training and test data are 0.99 and 0.94, respectively. The magnitude of the coefficient of determination (R^2) indicates the proportion of actual data variance of a predictor variable (here cumulative time-to-failure) that can be explained by the model. High R^2 values for the proposed model of the reliability estimation illustrate that 94% of the cumulative time-to-failure data variances is explained by the model. Paired *t*-statistics were also performed to check the statistical similarities between the actual values and the model-predicted reliability values. The *t*-statistics along with the significance level for the means of actual values and the model-predicted values for the training and test data set are presented in Table 5. Observe from the table that there is no significant difference in means for cumulative time-to-failure of training and test data. Fig. 10 shows scatterplots of actual values versus model-predicted values for the test data set.

To prove the superiority of the proposed method, a comparative study was carried out with different methods. The comparison was based on estimated values of test data. Table 6 presents the actual and predicted reliability values of the test data set. Observe from the table that the proposed method performed better in terms of minimizing the NRMSE than traditional neural networks and autoregressive models.

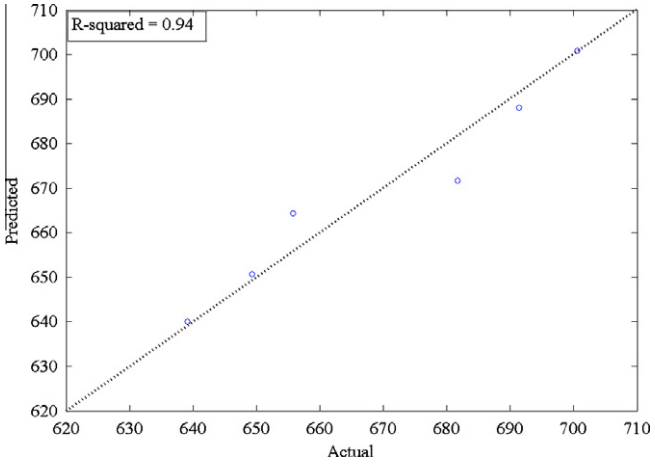

Fig. 10. Scatterplot of observed vs. predicted value of four attributes using neural regression.

5. Conclusions

We propose a genetic algorithm-based neural network (NN) model for forecasting systems reliability. The number of input variables for this NN model of reliability plays an important role. Results show that an entropy-based approach helps with optimum selection of the input parameters for the NN forecasting model. The main application of the genetic algorithm was for selection of the artificial NN parameters, that is, learning rate and momentum (η and μ). This study shows that selected parameters not only improve the performance of the model but also significantly reduce the computational time by eliminating the trial-and-error exercise.

We validated the proposed model using two benchmark data sets, and evaluated a comparative study of the predictive performance of various time series models. Results from benchmark studies reveal that the genetic algorithm-based NN with optimum input variable selection significantly improves the performance of the model. The importance of the proposed method is that no a priori specifications of parametric failure distributions need to be assumed. Results from the benchmark data sets show that the proposed method performs better than existing methods.

We conducted a case study of time-to-failure of a LHD machine. A comparison with other methods shows that the proposed method performs significantly better than existing models. The results of this research clearly demonstrate the potential of this approach for predicting failures and reliability of any system.

The main limitation of this study is that parameters related to a genetic algorithm, for example, crossover rate and mutation rate, are selected randomly. However, these parameters also play a role in development of the model. In the future, genetic algorithm parameters will be considered to improve the predictive performance of the model. Hidden node selection is also not an integrated part of genetic optimization. Future study will include incorporation of the hidden node number in the global search algorithm.

Appendix A

Data on turbochargers in a diesel engine.

Failure order number (i)	Reliability $R(T_i)$	Failure order number (i)	Reliability $R(T_i)$
1	0.993	21	0.7938
2	0.9831	22	0.7839
3	0.9731	23	0.7739
4	0.9631	24	0.7639
5	0.9532	25	0.754
6	0.9432	26	0.744
7	0.9333	27	0.7341
8	0.9233	28	0.7241
9	0.9133	29	0.7141
10	0.9034	30	0.7042
11	0.8934	31	0.6942
12	0.8835	32	0.6843
13	0.8735	33	0.6743
14	0.8635	34	0.6643
15	0.8536	35	0.6544
16	0.8436	36	0.6444
17	0.8337	37	0.6345
18	0.8237	38	0.6245
19	0.8137	39	0.6145
20	0.8038	40	0.6046

References

Amjady, N., & Ehsan, M. (1999). Evaluation of power systems reliability by artificial neural network. *IEEE Transaction on Power Systems, 14*, 287–292.

Bishop, M. (1998). *Neural networks for pattern recognition*. Oxford: Clarendon Press.

Box, G. E. P., & Jenkins, G. M. (1976). *Time series analysis: Forecasting and control*. San Francisco, CA: Holden-Day Inc..

Chang, P. T., Lin, K. P., & Pai, P. F. (2004). Hybrid learning fuzzy neural models in forecasting engine systems reliability. In *Proceedings of the fifth Asia Pacific industrial engineering and management systems conference, Gold Coast, Australia* (pp. 2361–2366).

Congdon, C. B. (1995). A comparison of genetic algorithm and other machine learning systems on a complex classification task from common disease research. PhD thesis, Computer Science and Engineering Division, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA.

Dengiz, B., Atiparmak, F., & Smith, A. E. (1997). Local search genetic algorithm for optimization of highly reliable communications networks. *IEEE Transactions on Evolutionary Computation, 1*(3), 179–188.

El-Sebakhy, E. A. (2009). Software reliability identification using functional networks: A comparative study. *Expert Systems with Applications, 36*(2), 4013–4020.

Fonseca, D. J., & Knapp, G. M. (2000). An expert system for reliability centered maintenance in the chemical industry. *Expert Systems with Applications, 19*(1), 45–57.

Hagan, M. T., Demuth, H. B., & Beale, M. (1995). *Neural network design*. Boston, MA: PWS Publishing Company.

Haykins, S. (1999). *Neural networks: A comprehensive foundation* (2nd ed.). New Jersey: Prentice Hall.

Ho, S. L., & Xie, M. (1998). The use of ARIMA models for reliability forecasting and analysis. *Computers and Industrial Engineering, 35*(1–2), 213–216.

Ho, S. L., Xie, M., & Goh, T. N. (2002). A comparative study of neural network and Box-Jenkins ARIMA modeling in time series forecasting. *Computers and Industrial Engineering, 42*, 371–375.

Holland, J. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: The University of Michigan Press.

Honarkhah, M., & Caers, J. (2010). Stochastic simulation of patterns using distance-based pattern modeling. *Mathematical Geosciences, 42*, 487–517.

Hu, C.-H., Si, X.-S., & Yang, J.-B. (2010). System reliability prediction model based on evidential reasoning algorithm with nonlinear optimization. *Expert Systems with Applications, 37*(3), 2550–2562.

Li, J.-P., Balazs, M. E., & Parks, G. T. (2007). Engineering design optimization using species-conserving genetic algorithms. *Engineering Optimization, 39*(2), 147–161.

Limas, M. C. (2006). A MORE flexible neural network package. <http://wiki.r-project.org/rwiki/doku.php?id=packages:cran:amore>.

Liu, M. C., Kuo, W., & Sastri, T. (1995). An exploratory study of a neural network approach for reliability data analysis. *Quality and Reliability Engineering International, 11*, 107–112.

Lolas, S., & Olatunbosun, O. A. (2008). Prediction of vehicle reliability performance using artificial neural networks. *Expert Systems with Applications, 34*(4), 2360–2369.

Lu, S., Lu, H., & Kloarik, W. J. (2001). Multivariate performance reliability prediction in real-time. *Reliability Engineering and System Safety, 72*, 39–45.

MacKay, D. J. C. (2003). *Information theory, inference, and learning algorithms*. Cambridge: Cambridge University Press.

Pai, P.-F. (2006). System reliability forecasting by support vector machine with genetic algorithms. *Mathematical and Computer Modeling, 43*, 262–274.

Su, C. T., Tong, L. I., & Leou, C. M. (1997). Combining time series and neural network approaches. *Journal of the Chinese Institute of Industrial Engineers, 4*, 419–430.

Wei, Z., Ma, D., Hong-Jun, Z., Wang, B.-L., & Yun-Tao, C. (2003). An application of multi-population genetic algorithm for optimization of adversaries' tactics and strategies in battlefield simulation. *International Conference on Machine Learning and Cybernetics, 3*(2–5), 1704–1707.

Willighagen, E. (2005). R based genetic algorithm for binary and floating point chromosomes. <http://rss.acs.unt.edu/Rdoc/library/genalg/html/00Index.html>.

Xu, K., Xie, M., & Tang, L. C. (2003). Application of neural networks in forecasting engine systems reliability. *Application of Soft Computation Journal, 2*(4), 255–268.

Yeh, W.-C., Lin, Y.-C., & Chung, Y. Y. (2010). Performance analysis of cellular automata Monte Carlo Simulation for estimating network reliability. *Expert Systems with Applications, 37*(5), 3537–3544.

Yi-Hui, L. (2007). Evolutionary neural network modeling for forecasting the field failure data of repairable systems. *Expert Systems with Applications, 33*(4), 1090–1096.

Zheng, J. (2009). Predicting software reliability with neural network ensembles. *Expert Systems with Applications, 36*(2), 2116–2122.