Expert Systems with Applications xxx (2008) xxx–xxx

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Software reliability identification using functional networks: A comparative study

Emad A. El-Sebakhy *

Information and Computer Science Department, College of Computer Science and Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

ARTICLE INFO

Available online xxxx

Keywords:
Software reliability
Functional networks
Minimum description length
Predictive models
Support vector machines
Neural networks

ABSTRACT

Software engineering development has gradually become essential element in different aspects of the daily life and an important factor in numerous critical real-industry applications, such as, nuclear plants, medical monitoring control, real-time military, bioinformatics, oil and gas industry, and air traffic control. This paper proposes a functional network as a novel computational intelligence scheme for tracking and predicting the software reliability. Several applications are presented to illustrate this new intelligent system framework models. To demonstrate the usefulness of functional networks and the existing data mining schemes, we briefly describe the learning algorithm of functional networks associativity model in predicting the software reliability. Comparative studies will be carried out to compare the performance of functional networks with the most popular existing data mining techniques, such as, statistical regression multilayer feed forward neural networks, and support vector machines. The results show that the performance of functional networks is more reliable, stable, accurate, and outperforms other techniques.

1. Introduction

Since computer software has become essential element in different aspects of our daily life and an important factor in numerous critical real-industry applications, there is a great demand for high-quality software products. One of the fundamental quality characteristics for the software is the reliability. Software reliability deals with behavior of a software system. It can be defined as the probability of a system to work without failures for a specified interval of time in a particular environment. It is possible to estimate the current or the future reliability of a system using reliability models. These models estimate the reliability based on failure data collected during the test phase. This, in turn, will help software project managers to estimate and schedule the time, effort, and cost to test and release the software at an acceptable level of reliability. There are many empirical correlations for modeling current and future reliability of a system. Different software reliability models were proposed to estimate and predict the reliability such as Aljahdali, Sheta, and Rine (2001), Cai, Cai, Wang, Yu, and Zhang (2001), Chen (2007), Costa, Vergilio, Pozo, and Souza (2005), El-Aroui and Soler (1996), Kimura, Yamada, and Osaki (1995), Lyu (1996), Musa, Jannino, and Okumoto (1987), Pai and Hong (2006), Pham and Pham (2000, 2001), Pham and Zhang (2003), Xie (1991), Xu, Xie, Tang, and Ho (2003) most of these modeling schemes were developed using linear or non-linear multiple regression, Bayesian statistical (BS), Autoregressive integrated moving average (ARIMA), multilayer feedforward neural networks (MFFN), radial basis function (RBF), genetic algorithm (GA), and support vector machines (SVMs). However, they often do not perform very accurate and suffer from a number of drawbacks. Each correlation was developed for a certain range of software reliability and geographical environment with similar failures for a specified interval. Thus, the accuracy of such correlations is critical and not often known in advance.

Recently, functional networks have been introduced by El-Sebakhy (2004), El-Sebakhy, Hadi, and Faisal (2007), El-Sebakhy, Faisal, El-Bassuny, Azzedin, and Al-Suhaim (2006), Li, Liao, Wu, and Yu (2001), Castillo (1998), Castillo, Cobo, Gutierrez, and Pruneda (1998), Castillo, Cobo, Gutierrez, and Hadi (1999), Castillo and Gutierrez (1998) as a generalization of the standard neural networks. It dealt with general functional models instead of sigmoid-like ones. In these networks the functions associated with the neurons are not fixed but are learnt from the available data. There is no need, hence, to include weights associated with links, since the neuron functions subsume the effect of weights. It is a general framework useful for solving a wide range of problems in probability, statistics, science, bioinformatics, biomedicine, structural civil engineering, signal processing, pattern recognition, functions approximations, real-time flood forecasting and other business and engineering applications (see El-Sebakhy, 2004; El-Sebakhy et al., 2006, 2007; Li et al., 2001; Bruen & Yang, 2005; Castillo, Cobo, & Gutierrez, 1998; Castillo, Gutiérrez, Hadi, & Lacruz, 2001; Castillo, Hadi, & Lacruz, 2001; Pruneda, Lacruz, & Solares, 2005; Rajasekaran, Mohan, & Khamis, 2004; Solares, Vieira, & Minguez, 2006) and the references therein for more details. The performance of functional networks has shown bright outputs for future applications in both industry and academic research of science and engineering based on its reliable and efficient results.

* Tel.: +966 3860 4263; fax: +966 3860 2175.
E-mail address: sebakhy@kfupm.edu.sa

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.02.053

Yet, this new intelligence system framework has not been utilized in the area of software engineering. The main contribution of this research is to propose functional networks as a new framework for predicting the software reliability. The built calibration model will be developed and tested based on the use of distinct known software reliability databases. Next, the developed functional networks model will be utilized to predict software reliability including time and frequency time between software successive failures.

The rest of this paper is organized as follows: Section 2 provides the related work and the literature review. The methodology and learning steps of functional networks as a new intelligence system paradigm is proposed in Section 3. Section 4 contains a brief explains of how the empirical study is conducted. The implantation process with parameters initialization is proposed in Section 5. Discussing both performance results and comparative studies were carried out in Section 6. The conclusion and recommendations for future works are drawn in Section 7.

2. Related work

Several statistics and data mining modeling schemes have been developed and applied for software reliability prediction. El-Aroui and Soler (1996) used *Bayesian statistical* model to predict software reliability. They made an assumption that the successive times between software failures follow the exponential distribution. The proposed model is useful for simulated software failure data based on the numerical examples. Pham and Pham (2000, 2001) applied Bayesian approach to predict software reliability. Their results show that proposed model outperforms other times-between-failures models in terms of the sum of square errors. Pham and Zhang (2003) proposed a prediction model based on *non-homogeneous Poisson process* (NHPP) approaches for software reliability forecasting. The results show that proposed model has better goodness-of-fit than other exist NHPP models.

Recently, machine learning techniques have been applied for predicting software reliability. A neural network was proposed by Cai et al. (2001) to predict software reliability. They evaluated the performance of neural network model by different network architectures. They concluded that neural network model is better for a smooth reliability data set than a fluctuating one. Xu et al. (2003) applied two techniques of neural networks: (i) multilayer perceptron feedforward neural networks and (ii) radial basis function to predict the reliability of engine systems. They found that the neural network approaches provide more accurate prediction results than the ARIMA model.

A multilayer perceptron feedforward neural network has been proposed by Aljahdali and Sheta (2001) as an alternative technique to build software reliability growth models. The proposed model was used with a slightly different configuration in which the number of neurons in the input layer represents the number of delay in the input data. They made a comparison between regression parametric models and neural network models. Their results indicate that neural networks were able to provide models with small sum of squares errors (SSE) than the regression model. Costa et al. (2005) used genetic programming (GP) to estimate the reliability growth. They carried out two experiments: one based on time and the other one based on test coverage. Also, they compared the results with other traditional and non-parametric ANN models.

Recently, used support vector machines (SVM) with simulated annealing algorithms to predict software reliability (Pai & Hong, 2006). The results indicate that the SVM model with simulated annealing algorithms provides more accurate prediction results than the other prediction models. Chen (2007) applied support vector regression with genetic algorithms to predict reliability in engine systems. The experimental results indicate that the proposed model outperforms the existing neural-network approaches and the ARIMA models based on the normalized root mean square error and mean absolute percentage error. Several other methods, such as Markov processes, Kalman filter model, operational-profile, and times-between-failures model had been developed in predicting software reliability (Kimura et al., 1995; Lyu, 1996; Musa et al., 1987; Xie, 1991). None of the above works, however, used and/or evaluated the capability of functional networks (FunNets) in predicting the software reliability growth.

3. Functional networks

3.1. Background

Recently, functional networks have been introduced by El-Sebakhy (2004), El-Sebakhy et al. (2007, 2006), Li et al. (2001), Castillo (1998), Castillo and Gutierrez (1998), Castillo et al. (1999, 1998) as a generalization of the standard neural networks. It dealt with general functional models instead of sigmoid-like ones. In these networks the functions associated with the neurons are not fixed but are learnt from the available data. There is no need, hence, to include weights associated with links, since the neuron functions subsume the effect of weights. Functional networks allow neurons to be multi-argument, multivariate, and different learnable functions, instead of fixed functions. Functional networks allow converging neuron outputs, forcing them to be coincident. This leads to functional equations or systems of functional equations, which require some compatibility conditions on the neuron functions. Functional networks have the possibility of dealing with functional constraints that are determined by the functional properties of the network model.

Functional networks as a new modeling scheme has been used in solving both prediction and classification problems. It is a general framework useful for solving a wide range of problems in probability, statistics, signal processing, pattern recognition, functions approximations, real-time flood forecasting, science, bioinformatics, medicine, structure engineering, and other business and engineering applications (see El-Sebakhy, 2004; El-Sebakhy et al., 2006, 2007; Li et al., 2001; Bruen & Yang, 2005; Castillo et al., 1998; Castillo, Hadi, et al., 2001; Pruneda et al., 2005; Rajasekaran et al., 2004; Solares et al., 2006; Castillo, Gutiérrez, et al., 2001) and the references therein for more details.

The performance of functional networks has shown bright outputs for future applications in both industry and academic research of science and engineering based on its reliable and efficient results. Several comparative studies have been carried to compare its performance with the performance of the most popular prediction/classification modeling data mining, machine learning schemes in the literature (El-Sebakhy, 2004). The results show that functional networks performance outperforms most of these popular modeling schemes in machine learning, data mining, and statistics communities (El-Sebakhy et al., 2006, 2007).

3.2. Functional networks in use: an example

Dealing with functional networks in prediction and classification required some concepts and definitions, which can be briefly summarized as follows:

Suppose that, we use the set $\mathbf{X} = \{x_1, \ldots, x_p\}$ to be the set of nodes, such that each node x_i is associated with a variable X_i. The neuron (neural) function over a set of nodes $\mathbf{X}$ is a tuple $U = \langle x, f, \mathbf{z} \rangle$,

Please cite this article in press as: El-Sebakhy, E. A. , Software reliability identification using functional networks: A comparative study, *Expert Systems with Applications* (2008), doi:10.1016/j.eswa.2008.02.053

where x is a set of the input nodes, f is a processing function and z is the output nodes, such that $z = f(x)$, where x and Z are two non-empty subsets of **X**. The best way to illustrate the functional networks and its way of representation is to use an example (El-Sebakhy et al., 2007). As it can be seen in Fig. 1, a functional network consists of: (a) several layers of storing units, one layer for containing the input data (x_i; $i = 1,2,3,4$), another for containing the output data (x_7) and none, one or several layers to store intermediate information (x_5 and x_6); (b) one or several layers of processing units that evaluate a set of input values and delivers a set of output values (f_i); and (c) a set of directed links. Generally, functional networks extend the standard neural networks by allowing neuron functions f_i to be not only true multiargument and multivariate functions, but to be different and learnable, instead of fixed functions. In addition, the neuron functions in functional networks are unknown functions from a given family, such as, polynomial, exponential, Fourier, to be estimated during the learning process. Furthermore, functional networks allow connecting neuron outputs, forcing them to be coincident (Castillo et al., 1999, 1998; El-Sebakhy et al., 2007).

The functional network uses two types of learning: (a) structural learning, (b) parametric learning. In structural learning, the initial topology of the network, based on some properties available to the designer is arrived at and finally a simplification is made using functional equation to a simpler architecture. In parametric learning, usually activation functions by considering the combination of "basis" functions are estimated by using the least square, steepest descent and mini-max methods (Cuddy, 1997). In this paper, we use the least square method for estimating activation functions.

Generally, functional network is a problem driven, which means that the initial architecture is designed based on a problem in hand. In addition, to the data domain, information about the other properties of the function, such as associativity, commutativity, and invariance, are used in selecting the final network. In the functional network, neuron functions are arbitrary known or unknown to be learned from the provided data, but in neural networks they are a sigmoid, linear or radial basis and other functions. In functional networks, neuron functions in which weights are incorporated are learned, and in neural networks, weights are learned. Neural networks work well if the input and output data are normalized in a specific range, such as, in between 0 and 1, but in functional networks there is no such restriction. Furthermore, it can be pointed out that neural networks are special cases of functional networks (El-Sebakhy et al., 2007). Dealing with functional networks required the several steps through the networks implementations and learning process.

3.3. Implementation and learning process in functional networks

First, define the problem in hand by specifying the initial topology based on the domain of the problem in hand. Second, simplify the chosen initial architecture using functional equations and the equivalence concept, then check the uniqueness condition of the desired architecture (see Castillo et al., 1998, 1999) for more details. Third, gather the required data and handle multicollinearity problem with the implementation of the required quality control check before the functional networks implementation. Fourth, the learning procedures and training algorithm based on either structure or parametric learning by considering the combinations of linear independent functions, $\Psi = \{\psi_{s1}, \ldots, \psi_{sm_s}\}$, for all s to approximate the neuron functions, that is

$$g_s(x) \doteq \sum_{i=1}^{m_s} a_{si}\psi_{si}(x) \quad \text{for all } s, \tag{1}$$

where the coefficients a_{si} are the parameters of functional networks. The most popular linearly independent functions in the literature are:

$$\Psi = \{1, X, \ldots, X^m\},$$

or

$$\Psi = \{1, \cos(X), \ldots, \cos^l(X), \sin^l(X)\}, \quad m = 2l,$$

or

$$\Psi = \{1, e^X, e^{-X}, \ldots, e^{mX}, e^{-mX}\},$$

where m is the number of elements in the combination of sets of linearly independent function. The parameters in (1) can be learned using one of the known optimization (loss criterion) techniques, such as, least squares estimation, conjugate gradient, iterative least squares, minimax, or maximum likelihood estimation. The last step in the implementation process is to select the best model and validate it. The best functional networks model is chosen based on the minimum description length and some other quality measurements, such as, the correlation coefficients and the root-mean-squared errors. The selection is achieved using one of the well known selection schemes, such as, exhaustive selection, forward selection, backward elimination, backward forward selection, and forward backward elimination. Therefore, if the validation performance is satisfactory, then the functional networks model is ready to be used in predicting unseen new unseen datasets from real-world industry applications.

The learning method of a functional network consists of obtaining the neuron functions based on a set of data $D = \{\mathbf{X}, \mathbf{Y}\}$, where $\mathbf{X} \in R^p$ represents the matrix of size $(n \times p)$ of p input feature variables and the column $Y \in R$ or $Y \subseteq \mathrm{R}$ for prediction or classification output results. The goal in both forecasting and classification is to determine the linear/nonlinear relationship among the output and the input variables using one of the following equations:

$$\begin{aligned} &g(y_i) = f(x_{i1}, \ldots, x_{ip}) + \varepsilon_i \quad \text{or} \\ &g(\pi_{ik}) = f(x_{i1}, \ldots, x_{ip}) + \varepsilon_i; \quad k = 1, \ldots, c. \end{aligned} \tag{2}$$

or

$$\begin{aligned} &y_i = f(x_{i1}, x_{i2}, \ldots, x_{ip}) + \varepsilon_i \quad \text{or} \\ &\pi_{ik} = f(x_{i1}, x_{i2}, \ldots, x_{ip}) + \varepsilon_i; \quad k = 1, \ldots, c. \end{aligned} \tag{3}$$

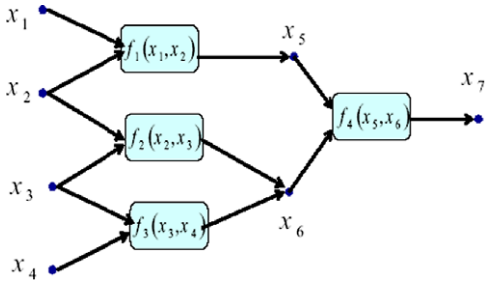

Fig. 1. Functional network topology: an example.

3.4. The most popular functional networks architectures in the literature

The most popular functional networks architecture used in literature are: the generalized associativity functional network (gafn) and the separable functional network (SFN) models to predict and identify the relationship functions $f(x_{i1}, \ldots, x_{ip})$. These two types of functional networks were described in details in Lyu (1996), El-Sebakhy et al. (2006). We briefly expressed these functional networks models as follows:

Please cite this article in press as: El-Sebakhy, E. A. , Software reliability identification using functional networks: A comparative study, *Expert Systems with Applications* (2008), doi:10.1016/j.eswa.2008.02.053

1. The *Generalized Associativity* model which leads to the additive model:

$$f(x_1,\ldots,x_k)=\sum_{j=1}^{k}h_k(x_k), \tag{4}$$

the corresponding architecture is shown in Fig. 2.

2. The *Separable functional networks model* which considers a more general form for the unknown function, $f(x_{i1},\ldots,x_{ip})$:

$$f(x_1,\ldots,x_k)=\sum_{r_1}^{q_1}\cdots\sum_{r_k}^{q_k}C_{r_1,\ldots,r_k}h_{r_1}(x_{r_1})\ldots h_{r_k}(x_{r_k}), \tag{5}$$

where $C_{r_1,\ldots,r_k}$ are unknown parameters and the sets of functions $\Phi_s=\{\phi_{r_s}: r_s=1,\ldots,q_s\}$, $s=1,2,...,k$, are linearly independent. An example of this functional network for $k=2$ and $q1=q2=q$ is shown in Fig. 3. Eqs. (4) and (5) are functional equations since their unknowns are functions. Their corresponding functional networks are the graphical representations of these functional equations.We note that the graphical structure is very similar to a neural network, but the neuron functions are unknown. Our problem consists of learning $h_1,h_2,\ldots,h_k$ in (4) and $C_{r_1,r_2,\ldots,r_k}$ in (5). In order to obtain $h_1,h_2,\ldots,h_k$ in (5), we approximate each $h_j(x_j)$, for $j=1,2,\ldots,k$ by a linear combination of sets of linearly independent functions ψ_{js} defined above, that is,

$$h_j(x_j)=\sum_{s=1}^{q_j}a_{js}\psi_{js}(x_j);\quad j=1,2,\ldots,p, \tag{6}$$

where the coefficients a_{js} subsume the actual parameters $C_{r_1,r_2,\ldots,r_k}$ in (5). Therefore, the problem is reduced to estimate the parameters a_{js}, for all j and s (El-Sebakhy et al., 2006; Lyu, 1996). Generally, by setting appropriate input and applying system identification to study the defect prone classes identification, one can customize the characteristics of input value according to wishful output. In this paper, we follow the same procedures in both (Lyu, 1996) and choose the least squares criterion to learn the parameters, but the additive model requires add some constrains to guarantee uniqueness. Alternatively, one can choose different optimization criterion based on his interest. The main advantage in choosing the least squares method is that the least squares criterion leads to solve a linear system of equations in both prediction and classification problems.

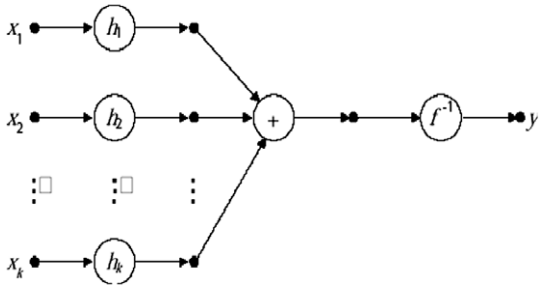

Fig. 2. Additive functional networks topology.

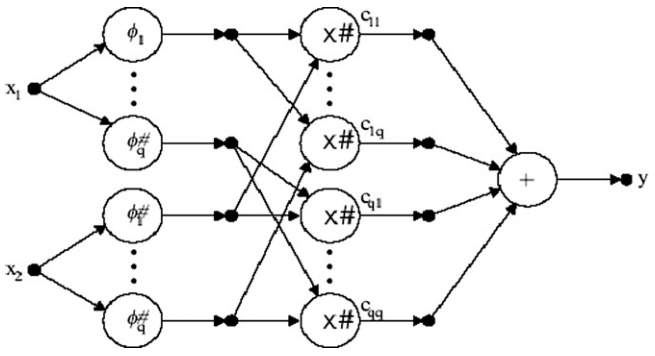

Fig. 3. The functional network for the separable model with $p=2$ and $q_1=q_2=q$.

In this research we utilized both types of functional networks to estimate the software reliability based on both actual inter-failure time and the accumulated frequency failure time. We follow the same procedures in both (Castillo et al., 1999; El-Sebakhy et al., 2007) and choose the least squares criterion to learn the parameters, but with the additive model requires add some constrains to guarantee uniqueness. Alternatively, one can choose different optimization criterion based on his interest. The main advantage in choosing the least squares method is that the least squares criterion leads to solve a linear system of equations in both prediction and classification problems.

4. Empirical study

This section describes the conducted empirical studies that are used in predicting the software reliability based on both actual inter-failure time and the accumulated frequency failure time data collected during the test phase. In this paper, we chosen only some of the most common utilized modeling schemes in software engineering, data mining, and machine learning communities, such as, multiple regression (MR), feedforward neural networks (FFN), and support vector regression (SVR) machines methods for the sake of simplicity and space to compare their performance with our developed functional networks model.

4.1. The main goal

The goal of this empirical study can be defined, using the GQM template (Basili & Rombach, 1988), as follows: *Compare* FunNets, SVM, MR, and FFN models *for the purpose of* software reliability prediction *with respect to* their prediction performance *from the point of view of* researchers and practitioners *in the context of* two software reliability datasets.

4.2. Characteristics of utilized datasets

Commonly, there are two types of failure data: time-domain data and interval-domain data. Time-domain data are reporting the individual times at which the failure occurred. Interval-domain data are counting the number of failures occurring during a fixed period of time. The time-domain data always provide better accuracy for the parameter estimates of current existing software reliability models. However, it involves more data collection efforts than the interval domain approach.

5. The implementations process and comparative studies

5.1. Initializations

To implement functional networks and the predictive models, the entire available dataset is usually divided into a training set and a testing or validating set. The training set is used to build the models and the testing set is used to evaluate the predictive capability of the models.

We use associativity functional networks model with only family of linearly independent families (basis) with polynomial degree at most 2. In addition, we implement *multiple regression* (MR), *feedforward neural networks* (FFN) based on both pure linear

and sigmoid activation neuron functions based on two/three hidden layers, *support vector regression* (SVR) *machines, and functional networks*. As it is commonly done in the literature (Duda, Hart, & Stock, 2001), we implement these intelligent systems modeling schemes with the required initialization process for their initial parameters, which can be summarized as follows:

Multiple regressions: They do not require any parameters to pass other than the method of optimization.

Support vector machines regression: This classifier requires several parameters. These are:

1. The kernel function (matrix), $\phi(x)$. The most common kernel functions used in the literature are: *Linear*, *Polynomial*, *Gaussian or RBF*, *Sigmoidal*, MLP, and *Fourier series*. Each one of these kernel functions has its own parameters. After so many trials to check which kernel gives the best performance in most situations, we found that the Sigmoidal kernel is the best kernel for support vector machines classifier.
2. The *parameters* of each kernel function have to be initialized. *Linear kernel* has no parameter; *polynomial kernel* has only one parameter, which is, q, the most common choices for q are: 3 or 5.
3. A *constant M* is commonly recommended value is 100.

Feedforward neural networks: It requires six parameters: the number of hidden layers, the number of nodes in each hidden layer, tolerance value for convergence, the initial weight, the maximum number of iterations, and the activation function. In the literature, multilayer Feedforward network is the most commonly used network with the backpropagation algorithm. It has been shown in the neural network literature that a one-hidden layer network may be adequate. The most common choice for the number of hidden nodes is $\min\{2p+2, n/10\}$, where p is the number of input nodes, and n is the total number of the observations in a given training database. Therefore, as network architecture, a three layered, fully connected, feedforward multi-layer perceptron (MLP) was used. The MLP networks often use the *log-sigmoid (logsig)* or *tan-sigmoid (tansig)* transfer function. Therefore, we use the transfer functions (*tansig* or *logsig*) and *purelin* for input to hidden, hidden to output. To run the feedforward neural networks classifier using the matlab built-in function *newff()*, we use 3000 for the maximum iteration, and a tolerance of 0.01.

Functional networks require the parameter, q, which is the degree of the approximating polynomial and the type of linearly independent family.

5.2. Implementation process

For the sake of both statistical regression and neural networks, the input datasets were normalized based on Eq. (7) to make sure that the utilized input variables were independent of measurement units. Thus, the predictors are normalized to interval of [0,1].

$$x_i^{(\text{new})} = \frac{(x_i^{(\text{old})} - \min(x_i))}{(\max(x_i) - \min(x_i))}; \quad i = 1, \ldots, n. \tag{7}$$

Through the implementation, the input parameters for each technique were chosen in such away to give the best performance. The results are summarized by computing both root mean-squared errors (RMSE) and correlation coefficient (R or R^2).

Based on the above explanation and the setting up parameters the implantation of the calibration functional networks model is built on first 70% of the provided data (learning model) and it was validated and tested using the remaining 30% of the provided data for testing set to test (external validate) the built model. The implantations are done using MATLAB V7 under Pentium *M* personal computer, and then compute the above discussed prediction performance measures, i.e. RMSE and R^2.

5.3. Statistical quality measures

In order to evaluate the performance of the compared prediction models, we considered two commonly used evaluative measures: correlation coefficient and root mean squared error (RMSE).

Correlation coefficient represents the degree of success in reducing the standard deviation by regression analysis, defined as

$$r = \sqrt{1 - \sum_{i=1}^{n} \left[(y)_{\text{exp}} - (y)_{\text{est}}\right]_i^2 / \sum_{i=1}^{n} \left[(y)_{\text{exp}} - \bar{y}\right]}, \tag{8}$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n}[(y)_{\text{exp}}]_i$ ranges from 1 for perfect positive correlation results, through 0 when there is no correlation, to −1 for perfect negative correlation.

Root Mean Squares Error: measures the data dispersion around zero deviation which is defined as

$$\text{RMSE} = \left[\frac{1}{n}\sum_{i=1}^{n} E_i^2\right]^{1/2} \tag{9}$$

where E_i is a relative deviation of an estimated value from an experimental input datasets:

$$E_i = \left[\frac{(y)_{\text{exp}} - (y)_{\text{est}}}{(y)_{\text{exp}}}\right] \times 100; \quad i = 1(1)n. \tag{10}$$

The best model is the one with the highest correlation coefficient and the smallest root of the mean-squared errors.

6. Results and discussion

For studying the performance of functional Networks, SVM, MLPFFN, and statistical regression based on the available data: AT&T Bell telemetry network system and real-time control system, details are shown in both Figs. 4 and 5. Each data set was divided into two separate sets namely 'training' and 'testing' using a suitable stratified sampling approach and cross-validation

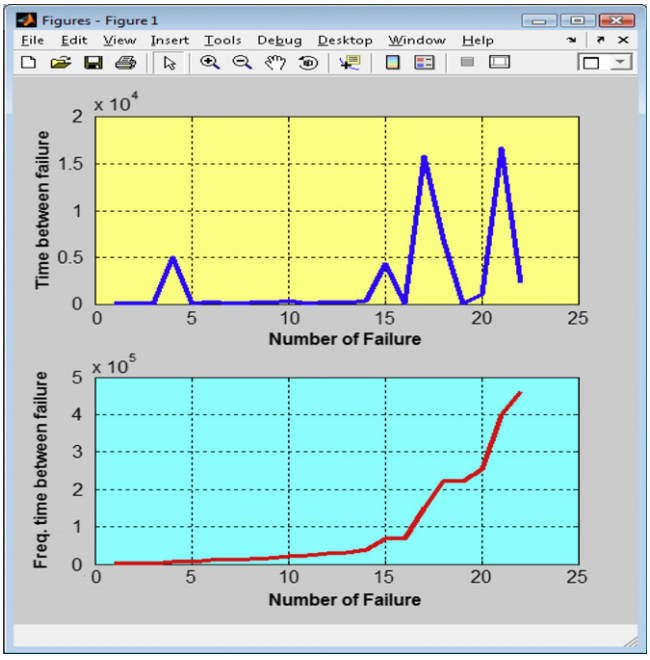

Fig. 4. Software reliability data from AT&T Bell telemetry network system.

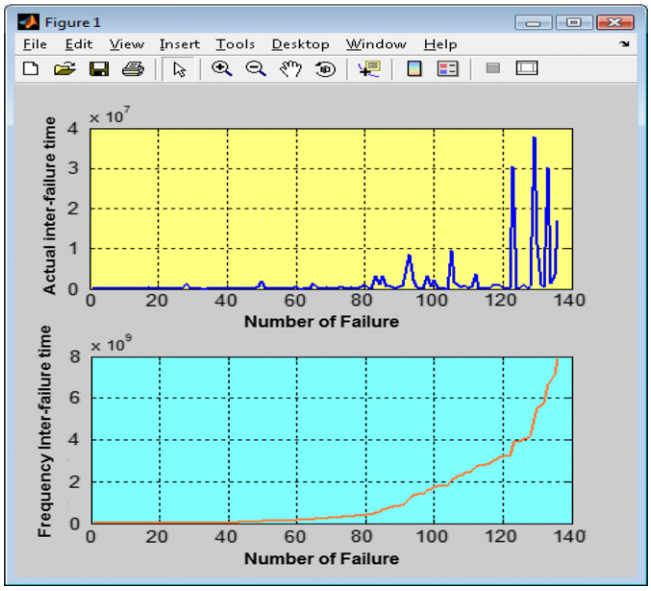

Fig. 5. Software reliability data from a real-time control system.

criterion. The data points were selected randomly for training and testing sets, such that, we randomly select 70% of the data for training to build the calibration model, while the remaining 30% was used for testing the built system. The criteria used for evaluating the predicted values include:

- The agreement between the predicted future accumulated reliability for software and the actual reliability for software.
- The correlation between the predicted and the actual accumulated reliability for software; and
- The RMSE (root mean square of error) between the actual and the predicted values.

In the current study, the performance of the functional network was compared with that of statistical regression (SR) and feedforward neural networks modeling techniques. The results obtained during both training and validation of the four modeling schemes were recorded when we used the two datasets. For the sake of both simplicity and space, we recorded only the plots that were corresponding to both functional networks and support vector machines as it is shown in Figs. 6–9. Tables 1 and 2 summarized the results of all the used computational intelligence modeling schemes based on both datasets: *AT&T Bell telemetry network system* and *real-time control system* in training and testing processes. The better results were shown in boldfaces.

As it can be seen in Table 1 that, the performances of all modeling schemes in the training process were very good and they are close to each other in the correlation coefficients, except multiple regression still the worst in both correlation coefficients and root-mean squared errors values. In addition, feedforward neural networks, support vector machines, and functional networks have the highest correlation coefficients, but support vector machine has the lowest root mean squared errors in the training process. On the other hand, in the testing and validation process, functional networks with the second-order polynomial degree have the highest correlation coefficient and the lowest root mean squared errors. The results show that functional networks is more accurate and reliable to predict the future software reliability in developing a new project, which it is shown a bright light for future use in different applications, such as, maintainability, risk analysis, and software cost estimation.

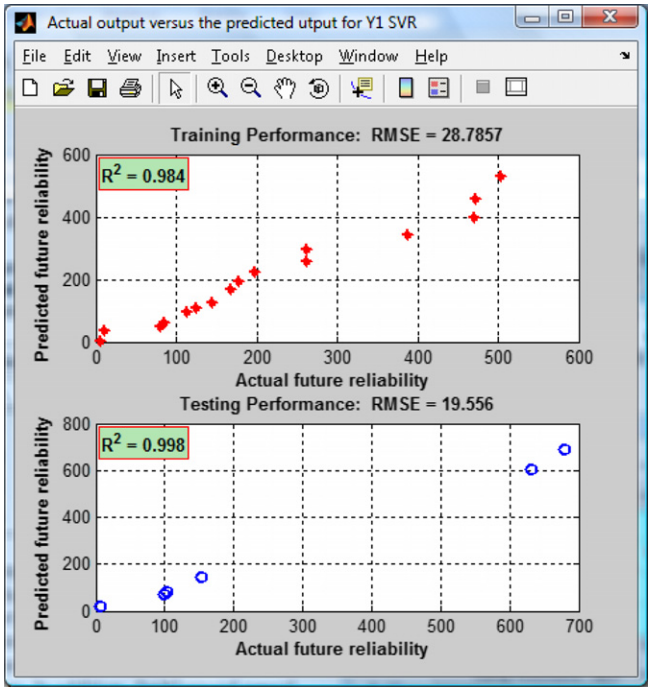

Fig. 6. Prediction of future reliability using support vector machines with $q = 2$ based on number of failures for the AT&T Bell telemetry network system.

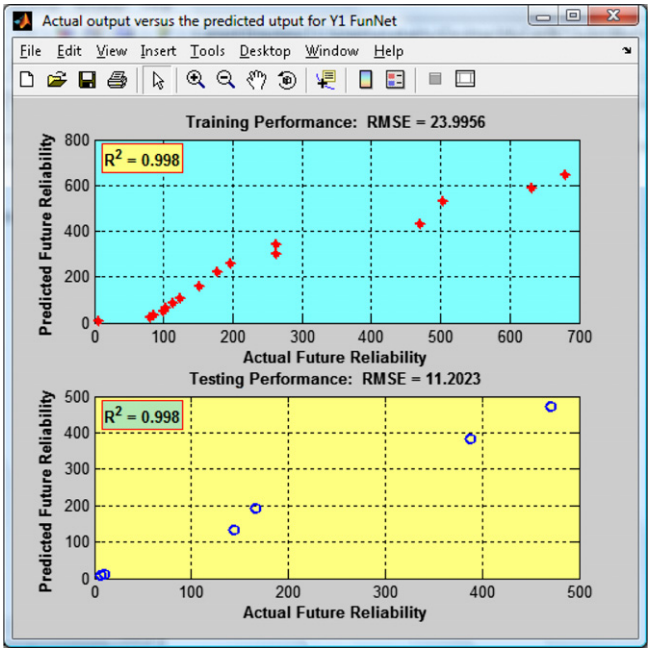

Fig. 7. Prediction of future reliability using functional networks with $q = 2$ based on number of failures for the AT&T Bell telemetry network system.

As it can be seen in Table 2 that, the performances of all modeling schemes were acceptable in the training process, still multiple regression has the worst performance in both correlation coefficients and root-mean squared errors values. Functional networks have the highest correlation coefficients and lowest root mean squared errors in both training and testing process. Furthermore, the results support the previous obtained performance of functional networks, which gives an indicator that; functional net-

Please cite this article in press as: El-Sebakhy, E. A. , Software reliability identification using functional networks: A comparative study, *Expert Systems with Applications* (2008), doi:10.1016/j.eswa.2008.02.053

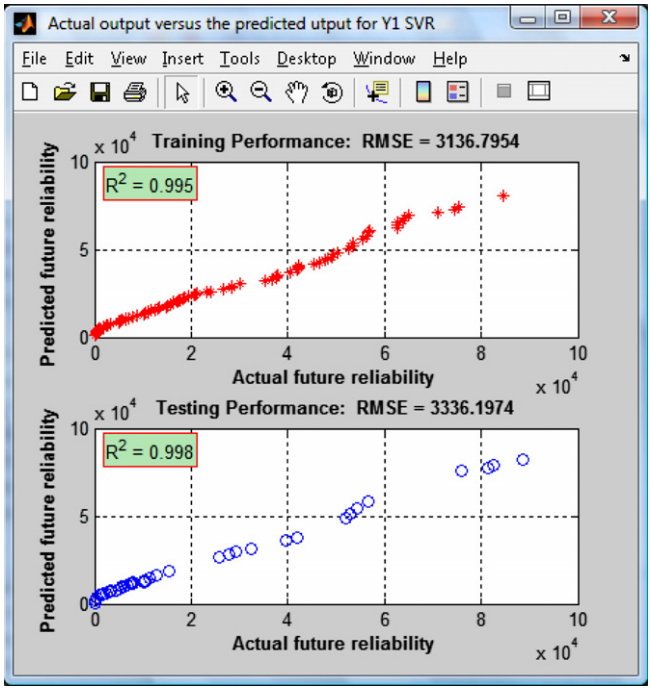

Fig. 8. Prediction of future reliability using support vector machines regression based on number of failures for the Real-time control system.

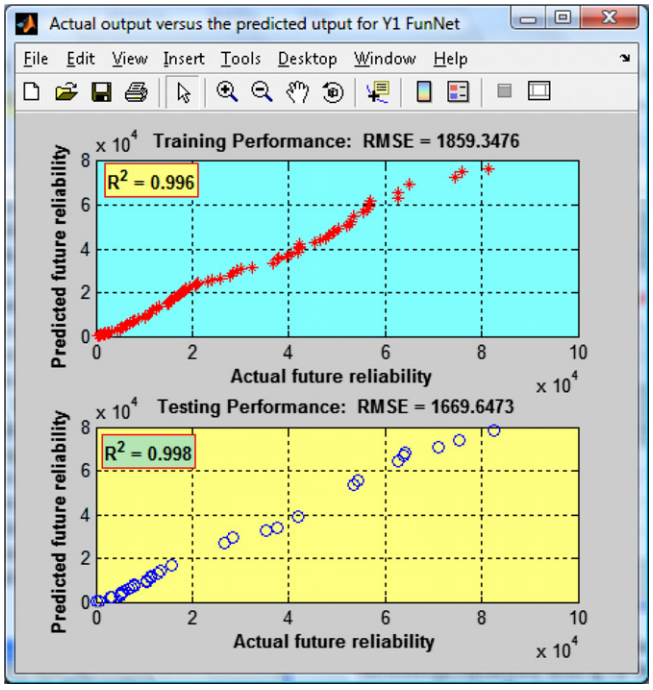

Fig. 9. Prediction of future reliability using functional networks with $q = 2$ based on number of failures for the Real-time control system.

works have a reliable and efficient prediction for the future software reliability in developing a new project. We conclude that functional networks are efficient reliable modeling technique that is trustable to be used in the area of software engineering. It is shown a bright light performance for future use in different applications, such as, maintainability, risk analysis, and software cost estimation.

Table 1
The correlation coefficient and RMSE values for the used four computational intelligence modeling schemes based on AT&T Bell telemetry network system

Model	Training		Testing	
	R^2	RMSE	R^2	RMSE
MR	0.932	51.83	0.988	132.29
FFN	**0.984**	29.92	0.996	50.47
SVM	**0.982**	**28.79**	0.996	19.56
FunNets	**0.998**	24	**0.998**	**11.2**

Table 2
The correlation coefficient and RMSE values for the used four computational intelligence modeling schemes based on real-time control system

Model	Training		Testing	
	R^2	RMSE	R^2	RMSE
MLR	0.935	7838.41	0.9611	8132.59
ANN	0.9963	2112.73	0.9973	1697.91
SVR	0.9948	3136.8	0.9978	3336.2
FunNet	**0.9963**	**1859.35**	**0.998**	**1669.65**

7. Conclusion

Overall, the results in both implementations shown that the performances of all modeling schemes were acceptable in the training process; still multiple regressions has the worst performance in both correlation coefficients and root-mean squared errors values. Functional networks have the highest correlation coefficients and lowest root mean squared errors in both training and testing process. Furthermore, the results support the previous obtained performance of functional networks, which gives an indicator that; functional networks have a reliable and efficient prediction for the future software reliability in developing a new project. We conclude that functional networks are efficient reliable modeling technique that is trustable to be used in the area of software engineering. It is shown a bright light performance for future use in different applications, such as, maintainability, risk analysis, and software cost estimation.

We suggest for future work to use different independent families other than polynomial and use more databases for software engineering quality assurance. We should try the neuro-fuzzy inference systems as well as extreme learning machine during the comparative studies.

Acknowledgments

The author would like to thank Mansoura University, Egypt and King Fahd University of Petroleum and Minerals, Saudi Arabia for the facilities utilized to perform the present work and for their support.

References

Aljahdali, S., Sheta, A., & Rine, D. (2001). Prediction of software reliability: A comparison between regression and neural network non-parametric models. In *Proceedings of the ACS/IEEE international conference on computer systems and applications*.

Basili, V., & Rombach, H. (1988). The TAME project: Towards improvement-oriented software environment. *IEEE Transactions on Software Engineering, 14*, 758–773.

Bruen, M., & Yang, J. (2005). Functional networks in real-time flood forecasting – A novel application. *Advances in Water Resources, 28*, 899–909.

Cai, K., Cai, L., Wang, W., Yu, Z., & Zhang, D. (2001). On the neural network approach in software reliability modeling. *The Journal of Systems and Software, 58*, 47–62.

Castillo, E. (1998). Functional networks. *Neural Processing Letters, 7*(3), 151–159.

Castillo, E., & Gutierrez, J. M. (1998). A comparison of functional networks and neural networks. In *Proceedings of the IASTED international conference on artificial intelligence and soft computing* (pp. 439–442), IASTED ACTA Press.

Please cite this article in press as: El-Sebakhy, E. A. , Software reliability identification using functional networks: A comparative study, *Expert Systems with Applications* (2008), doi:10.1016/j.eswa.2008.02.053

Castillo, E., Cobo, A., & Gutierrez, J. M. (1998). Nonlinear time series modeling and prediction using functional networks. Extracting information masked by chaos. *Physical Letters A, 244*, 71–84.

Castillo, E., Cobo, A., Gutierrez, J. M., & Hadi, A. S. (1999). A general framework for functional networks. *Networks, 35*, 70–82.

Castillo, E., Cobo, A., Gutierrez, J. M., & Pruneda, E. (1998). *Functional networks with applications-a neural-based paradigm*. Boston/Dordrecht/London/New York: Kluwer Academic Publishers.

Castillo, E., Gutiérrez, J. M., Hadi, A. S., & Lacruz, B. (2001). Some applications of functional networks in statistics and engineering. *Technometrics, 43*, 10–24.

Castillo, E., Hadi, A., & Lacruz, B. (2001). Optimal transformations in multiple linear regression using functional networks. In *Proceedings of the international work-conference on artificial and natural neural networks*. IWANN 2001, in Lecture Notes in Computer Science 2084, Part I, 2001; 316–324.

Chen, K. (2007). Forecasting systems reliability based on support vector regression with genetic algorithms. *Reliability Engineering and System Safety, 92*, 423–432.

Costa, E., Vergilio, S., Pozo, A., & Souza, G. (2005). Modeling software reliability growth with genetic programming. In *Proceedings of the 16th IEEE international symposium on software reliability engineering (ISSRE'05)*.

Cuddy, S. (1997). The application of mathematics of fuzzy logic to petrophysics. *SPWLA Annual Logging Symposium*.

Duda, R. O., Hart, P. E., & Stock, D. G. (2001). *Pattern classification* (second ed.). New York: John Wiley and Sons.

El-Aroui, M., & Soler, J. (1996). A Bayes nonparametric framework for software-reliability analysis. *IEEE Transactions on Reliability, 45*, 652–660.

El-Sebakhy, A. E. (2004). Functional networks training algorithm for statistical pattern recognition; IEEE symposium on computers and communications, ISCC 2004. *Ninth International Symposium on Volume, 1*, 92–97.

El-Sebakhy, E., Faisal, K., El-Bassuny, T., Azzedin, F., Al-Suhaim, A. (2006). Evaluation of breast cancer tumor classification with unconstrained functional networks classifier. In *Proceeding of the 4th ACS/IEEE international conference on computer systems and applications* (pp. 281–287).

El-Sebakhy, A. E., Hadi, S. A., & Faisal, A. K. (2007). Iterative least squares functional networks classifier. *IEEE Transactions Neural Networks, 18*(2), 844–850.

Kimura, M., Yamada, S., & Osaki, S. (1995). Statistical software reliability prediction and its applicability based on mean time between failures. *Mathematical and Computer Modeling, 22*, 149–155.

Li, C., Liao, X., Wu, Z., & Yu, J. (2001). Complex functional networks. *Mathematical Computer Simulation, 57*, 355–365.

Lyu, M. (1996). *Handbook of software reliability engineering*. New York: McGraw-Hill.

Musa, J., Jannino, A., & Okumoto, K. (1987). *Software reliability measurement, prediction, application*. New York: McGraw-Hill.

Pai, P., & Hong, W. (2006). Software reliability forecasting by support vector machines with simulated annealing algorithms. *The Journal of Systems and Software, 79*, 747–755.

Pham, L., & Pham, H. (2000). Software reliability models with time dependent hazard function based on Bayesian approach. *IEEE Transactions on Systems, Man, and Cybernetics Part A, 30*, 25–35.

Pham, L., & Pham, H. (2001). A Bayesian predictive software reliability model with pseudo-failures. *IEEE Transactions on Systems, Man, and Cybernetics Part A, 31*, 233–238.

Pham, H., & Zhang, X. (2003). NHPP software reliability and cost models with testing coverage. *European Journal of Operational Research, 145*, 443–454.

Pruneda, S., Lacruz, B., & Solares, C. (2005). *A first approach to solve classification problems based on functional networks. ICANN 2005, LNCS 3697*. Berlin Heidelberg: Springer-Verlag (pp. 313–318).

Rajasekaran, S., Mohan, V. S., & Khamis, O. (2004). The optimization of space structures using evolution strategies with functional networks. *Engineering with Computers, 20*, 75–87.

Solares, C., Vieira, E. W., & Minguez, R. (2006). Functional networks and the lagrange polynomial interpolation. In E. Corchado (Ed.), *IDEAL 2006, LNCS 4224* (pp. 394–401). Berlin Heidelberg: Springer-Verlag.

Xie, M. (1991). *Software reliability modeling*. Singapore: World Scientific.

Xu, K., Xie, M., Tang, L. C., & Ho, S. L. (2003). Application of neural networks in forecasting engine system reliability. *Applied Soft Computing, 2*(4), 255–268.

Please cite this article in press as: El-Sebakhy, E. A. , Software reliability identification using functional networks: A comparative study, *Expert Systems with Applications* (2008), doi:10.1016/j.eswa.2008.02.053