Empir Software Eng (2012) 17:738–756
DOI 10.1007/s10664-011-9192-6

Software development effort prediction of industrial projects applying a general regression neural network

Cuauhtemoc Lopez-Martin · Claudia Isaza · Arturo Chavoya

Published online: 3 December 2011

Editors: Martin Shepperd

Abstract An important factor for planning, budgeting and bidding a software project is prediction of the development effort required to complete it. This prediction can be obtained from models related to neural networks. The hypothesis of this research was the following: effort prediction accuracy of a general regression neural network (GRNN) model is statistically equal or better than that obtained by a statistical regression model, using data obtained from industrial environments. Each model was generated from a separate dataset obtained from the International Software Benchmarking Standards Group (ISBSG) software projects repository. Each of the two models was then validated using a new dataset from the same ISBSG repository. Results obtained from a variance analysis of accuracies of the models suggest that a GRNN could be an alternative for predicting development effort of software projects that have been developed in industrial environments.

Keywords Software development effort prediction · General regression neural network · Statistical regression · ISBSG · Repeatability

1 Introduction

Bad effort estimates may lead to poor planning, low profitability and consequently, products with poor quality (Jørgensen 2006). Underestimating a project may lead to under-staffing it, under-scoping the quality assurance effort, and setting short schedules.

C. Lopez-Martin (✉) · A. Chavoya
Information Systems Department CUCEA, Guadalajara University, P.O. Box 45100, Jalisco, Mexico
e-mail: cuauhtemoc@cucea.udg.mx

A. Chavoya
e-mail: achavoya@cucea.udg.mx

C. Isaza
Department of Electronic Engineering-GEPAR Research Group, Universidad de Antioquia, Medellín, Colombia
e-mail: cisaza@udea.edu.co

Overestimating a project could lead to assigning more resources than it really needs, the project is then likely to cost more than it should, take longer to deliver than necessary, delay the use of the resources in the next project (De Barcelos et al. 2008), and possibly the project could not win in a competitive bid scenario.

Software development estimation techniques can be classified into two general categories:

1) Expert judgment. This technique implies a lack of analytical argumentation and aims at deriving estimates based on the experience of experts on similar projects; this technique is based on a tacit (intuitive) quantification step (Jørgensen 2007b).
2) Model-based technique. It is based on a deliberate (mechanical) quantification step, and it could be divided into the following two subcategories:
   a) Models based on Statistics: Its general form is a statistical regression model (Boehm et al. 2000).
   b) Models based on computational intelligence: These techniques include fuzzy logic (López-Martín 2011a), neural networks (De Barcelos et al. 2008), genetic programming (Afzal and Torkar 2011), and genetic algorithms (Sun-Jen et al. 2008).

Based on the assumption that no single technique is best for all situations and that a careful comparison of the results of several approaches is more likely to produce realistic estimates (Boehm et al. 2000), this study compares the following two models against each other: statistical regression and a type of neural network termed general regression neural network (GRNN). A GRNN was selected as it was found to yield an acceptable accuracy when applied for predicting the software development effort of individual projects (López-Martín 2011b). The comparison against a statistical regression model is made because a regression analysis should be done as the default model construction method (Kitchenham et al. 2007), which has been the dominating model in recent years (Jørgensen and Shepperd 2007). These two models were generated from large projects. In order to obtain a representative model based on a pattern recognition method such as a GRNN, it is necessary to train the system with the most representative data (Duda et al. 2001). Therefore, data samples for this study have been carefully selected from a statistically significance analysis. A weakness of previous studies related to the use of the datasets is that the data quality is frequently not known or discussed in the corresponding publications (Jørgensen and Shepperd 2007).

Data from large projects were obtained from the International Software Benchmarking Standards Group (ISBSG) software projects repository; the use of this kind of dataset is important because a lot of studies are based on datasets that are clearly too old (such as the COCOMO 81 dataset) to be representative of more recent or future projects (Jørgensen and Shepperd 2007).

Comparison of models is based upon the two following main stages when an estimation model is used (Montgomery and Peck 2001):

1) The model adequacy checking or model verification (estimation) must be determined, that is, whether the model is adequate to describe the observed (actual) data; if so, then
2) The model is validated using new data (prediction).

In the field of neural networks, the first stage is commonly known as *training*, whereas the second stage is termed *testing*.

The hypothesis to be investigated in this paper is the following:

Effort prediction accuracy of a general regression neural network is statistically equal or better than that obtained by statistical regression, when data are obtained from industrial environments.

The remainder of this introduction section describes how each kind of project is measured, that is, their dependent and independent variables are described, the criterion for evaluating the models is described and justified, a brief introduction to neural networks in general and general regression neural networks in particular is described; the section finishes with the presentation of related work. The next section is dedicated to the description of the data sample. The following two sections correspond to the generation and the validation of the models respectively, whereas comparisons among models are presented in the next section. Finally, sections of discussion, conclusions and future research are presented.

1.1 Software Project Measurement

The effort of large projects was measured in man-months. The ISBSG dataset measures effort value in hours, which was converted to man-months having as equivalence 152 h per one man-month (Boehm et al. 2000).

Project size in the ISBSG dataset is mainly measured in adjusted function points (the number of lines of code by project was only available for some of the projects). Along with source lines of code, function points (FP) is one of the most common measures for estimating software project size, which is done by quantifying the amount of functionality provided to the user in terms of the number of inputs, outputs and files (Sheetz et al. 2009).

The function points counting process involves the following concepts:

1. Record Element Type (RET). A subgroup of data elements within an either internal or external file.
2. File Type Referenced (FTR). A file type referenced by a transaction (i.e. add, delete or modify data).
3. Data Element Type (DET). A unique user recognizable, non-recursive (non-repetitive) field.

The process for counting the function points is based on the following steps (Garmus and Herron 1996):

1) Identify data functions and their complexity. There are two types of data: internal and external.

   a) Internal Logical File (ILF). A user identifiable group of logically related data that resides entirely within the application boundary and is maintained through External Inputs. Even though it is not a rule, an ILF should have at least one external input. The ILFs are rated according to Table 1.
   b) External Interface File (EIF). A user identifiable group of logically related data that is used for reference purposes only. The data resides entirely outside the application boundary and is maintained by external inputs of another application. Each EIF must have at least one external output or external interface file. At least one transaction, external input, external output or external inquiry should include the EIF as an FTR. The EIFs are rated according to Table 1.

Table 1 Complexity matrix for ILF and EIF

RET	DET		
	1–19	20–50	Greater than 50
1	Low	Low	Average
2 to 5	Low	Average	High
6 or more	Average	High	High

2) Identify transactional function and their complexity. There are three transactional function types:

   a) External input (EI). An elementary process of the application that processes data or control information that enters from outside the boundary of the application, such as adding, changing, and deleting transactions. The external inputs are rated according to Table 2.
   b) External output (EO). An elementary process in which derived data passes across the boundary from inside to outside. Derived data occurs when one or more data elements are combined with a formula to generate or derive an additional data element. The external inputs are rated according to Table 3.
   c) External inquiry (EQ). An elementary process with both input and output components that result in data retrieval from one or more ILF and EIF. The input process does not update or maintain any FTR (ILF or EIF) and the output side does not contain derived data. The external inquiries are rated according to Table 3.

3) Determine the Unadjusted Function Points (UFP) count. The levels of ILF, EIF, EI, EO and EQ obtained from Tables 1, 2 and 3 are converted to a quantitative value according to Table 4. The final UFP are obtained by adding all these values up.
4) Determine the value adjustment factor (VAF) from 14 characteristics. The value adjustment factor (VAF) is obtained from 14 system characteristics that rate the general functionality of the application (Table 5). Each characteristic has associated a description as well as a degree of influence, whose range is from 0 to 5 having the following meaning:

   0 Not present, or no influence
   1 Incidental influence
   2 Moderate influence
   3 Average influence

Table 2 Functional complexity matrix for EI

FTR	DET		
	1–4	5–15	Greater than 15
Less than 2	Low	Low	Average
2	Low	Average	High
Greater than 2	Average	High	High

Table 3 Functional complexity matrix for EO and EQ

FTR	DET		
	1–5	6–19	Greater than 19
Less than 2	Low	Low	Average
2–3	Low	Average	High
Greater than 3	Average	High	High

4 Significant influence
5 Strong influence throughout

To reduce the bias, each degree of influence by characteristic has a specific meaning (Garmus and Herron 1996):

The VAF is obtained by using the following equation:

$$VAF = 0.65 + \left[\left(\sum_{i=1}^{14} C_i\right)/100\right]$$

Where:

C_i Degree of influence by the i_{th}-characteristic

5) Calculate the Adjusted Function Point (AFP) count by multiplying the UFP by the VAF.

1.2 Criterion for Evaluating Estimation Model

Considering that the accuracy indicator has a large impact on the results (Myrtveit et al. 2005), the selection of the accuracy criterion for this study was based on Foss et al. (2003). The Magnitude of error Relative to the Estimate (MER) is used as criterion for evaluating and comparing the estimation models of this study. The MER was selected because the Magnitude of Relative Error (or MRE, which has commonly been used as criterion) is not strongly recommended and MER has shown better results than MRE (Foss et al. 2003). The accuracy of an estimation technique is inversely proportional to the MMER. The MER for observation *i* is defined as follows:

$$MER_i = \frac{|\text{Actual Effort}_i - \text{Estimated Effort}_i|}{\text{Estimated Effort}_i}$$

Table 4 Unadjusted function points

Components	Function levels		
	Low	Average	High
Internal Logical File (ILF)	7	10	15
External Interface File (EIF)	5	7	10
External input (EI)	3	4	8
External output (EO)	4	5	7
External inquiry (EQ).	3	4	6

Table 5 System characteristics

Description
1. Data communications
2. Distributed data processing
3. Performance
4. Heavily used configuration
5. Transaction rate
6. On-Line data entry
7. End-user efficiency
8. On-Line update
9. Complex processing
10. Reusability
11. Installation ease
12. Operational ease
13. Multiple sites
14. Facilitate change

The MER value is calculated for each observation i whose effort is estimated. The aggregation of the MER over multiple observations (N) can be achieved through the Mean MER (MMER) as follows:

$$MMER = (1/N)\sum_{i=1}^{N} MER_i$$

This MMER criterion is in accordance with Kitchenham et al. (2007), who suggest that statistical tests based on the absolute residuals of the raw data should be performed.

In several papers, a MMRE≤0.25 has been considered as acceptable; however, the authors who have proposed this value present neither any reference to studies nor any argumentation providing evidence (Jørgensen 2007a). On the other hand, a reference for an acceptable value of MMER has not been found.

1.3 Neural Networks and General Regression Neural Networks

An artificial neural network, or more succinctly a *neural network* (NN), is a massively parallel, distributed system composed of simple processing units or artificial neurons that are interconnected to mimic a biological neural network (Haykin 1999). An artificial neuron or node in the network can have as input some data fed to the system or the outputs from other nodes. A neuron typically performs the summation of the product of input and weighted factors, and the resulting value is used as input to an activation function, which is usually sigmoid. The output from the activation function, which is actually the output from the neuron, can act as an excitatory or an inhibitory signal for other nodes, or it can be the output from the system.

The topology (the number, distribution and interconnection of nodes) of NNs can vary widely. One of the most used topologies is the multi-layered feedforward network, where there the nodes in the input layer take the data from outside the system and the nodes from the output layer provide the system's output. Between the input and output layers, there can be so-called hidden layers, where all nodes take as input the output from the previous layer and their output is fed to the inputs of the next layer (hence the name *feedforward*).

Before a NN can be used, it has to undergo some training, which involves iteratively finding the appropriate weight values so that the network outputs the desired value for a given a set of input values. A number of training algorithms have been developed over the years, with backpropagation being the most widely known. After a NN has been trained, it is convenient to validate its performance using ideally a dataset different from the one used to train it.

The General Regression Neural Network (GRNN) is a type of NN that performs regression on a continuous output variable and it has the following advantages: (a) fast learning and (b) convergence to the optimal regression surface as the number of samples becomes very large. The GRNN has shown that, even with sparse data in a multidimensional measurement space, the algorithm provides smooth transitions from one observed value to another (Specht 1991). Figure 1 shows the architecture of a GRNN (Specht 1991). The input units are merely distribution units, which provide all the scaled measurement variables X to either all neurons on the second layer, the pattern units that are dedicated to one exemplar, or one cluster center. When a new vector X is entered into the network, it is subtracted from the stored vector representing each cluster center. Either the squares or the absolute values of the differences are summed and fed into a nonlinear activation function. The activation function normally used is the exponential function. The pattern units output are passed onto the summation units. The summation units perform a dot product between a weight vector and a vector composed of the signals from the pattern units. The summation unit that generates an estimate of $F(X)K$ sums the outputs of the pattern units weighted by the number of observations each cluster center represents. The summation unit that estimates $Y'F(X)K$ multiplies each value from a pattern unit by the sum of the samples Y^j associated with cluster center X^i. The output unit merely divides $Y'F(X)K$ by $F(X)K$ to yield the desired estimate of Y. When estimation of a vector Y is desired, each component is estimated using one extra summation unit, which uses as its multipliers sums of samples of the component of Y associated with each cluster center X^i. Figure 1 shows a network that can be used to estimate a vector Y from a measurement vector X.

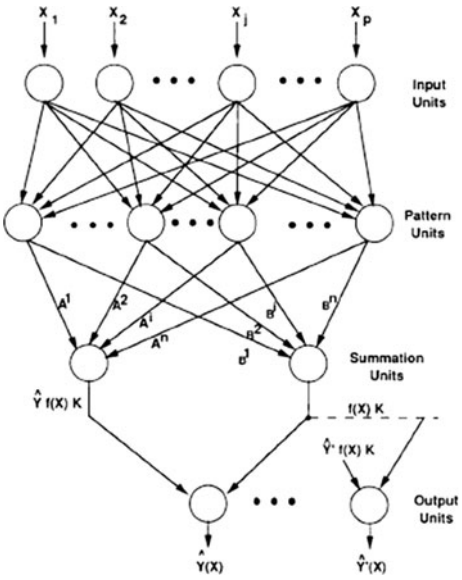


Fig. 1 General regression neural network diagram

1.4 Related Work

Paliwal and Kumar (2009) have identified the following problems in studies involving neural networks and statistical techniques that have been used for prediction in various areas of applications:

1) Most of the articles seem to have not used the statistical techniques optimally, including validity of assumptions. Violations of assumptions and comparison of neural networks with such techniques probably would have resulted in better performance.
2) The determination of various parameters associated with neural networks is not straightforward and finding the optimal configuration of neural networks is a very time consuming process.
3) The results obtained from the model building processes are not validated on a new dataset that is not used for building the models. Testing on separate dataset would have provided unbiased estimate of the generalization error.
4) It is not clear whether a statistically significant difference exists in the performance of different techniques that are compared.

In addition, some of the studies found by Paliwal and Kumar (2009) make use of statistical techniques to select the significant variables to get a better predictive or classification model.

With regard to the application of neural networks for predicting the development effort on software projects, we found the following studies:

Heiat (2002) compared the prediction accuracy of two kinds of neural networks (a multilayer perceptron and a radial basis function network) with the accuracy of a statistical regression. Three sets of data were used: the IBM DP Services Organization comprising 24 projects, the Kemerer dataset comprising 15 projects, and the Hallmark dataset comprising 28 projects. Heiat conducted two experiments, the projects from the Kemerer and the IBM datasets—which include third generation programming languages—were combined for the first experiment, whereas in the second experiment, projects from the Kemerer, the IBM, and the Hallmark datasets—which include both third generation and fourth generation programming languages—were combined. The three datasets were referenced in papers published before the year 1988. The results from this study indicated that when a combined third generation and fourth generation language dataset were used, the neural network produced improved performance over conventional statistical regression. The prediction accuracy evaluation criterion was the MMRE. The training set for the models included 32 projects for the first experiment and 60 for the second experiment, whereas seven projects were used for testing the models. Heiat suggested further studies with larger datasets to verify the results obtained in his study.

Oliveira (2006) provided a comparative study on support vector regression (SVR), a radial basis function neural network (RBFN) and linear regression. His result was based on a set of 18 projects developed before 1981 and it showed that SVR significantly outperforms RBFN and linear regression. The accuracy criterion was the Magnitude of Relative Error (MMRE).

Vinay et al. (2008) proposed the use of a wavelet neural network (WNN). The WNN is compared with a multilayer perceptron, a radial basis function network, multiple linear regression, a dynamic evolving neuro-fuzzy inference system, and a support vector machine (SVM). The accuracy criterion was the Magnitude of Relative Error (MMRE).

The two datasets used were from Canadian financial, and IBM data processing services (IBMDPS), having 24 and 37 software projects, respectively. These projects were developed in 1996 or before. Based on their experiments, it was observed that the WNN outperformed all the other techniques.

Park and Baek (2008) proposed and evaluated a neural network. They obtained their conclusions from a sample of 148 projects completed between 1999 and 2003 including a wide range of software development technologies, project scales and project periods, in addition to having projects covering various industries. The prediction accuracy evaluation criterion was the MMRE. They trained and applied three neural networks varying them in their independent variables: the first network had only function points (FPs) as variable, the second one had six variables excluding FPs, and the third one had seven variables: FPs plus the above-mentioned six variables. They concluded that the neural network model that used FPs and the six other input variables was superior to the others.

De Barcelos et al. (2008) compared the accuracy of a feedforward multilayer perceptron neural network against statistical regression. The COCOMO database of 63 projects, which was published in the year 1981, was used for training and testing the models. The prediction accuracy evaluation criterion was the MMRE. The focus of this study was on the investigation of the behavior of these two techniques when predicting variables as categorical variables are used. The independent variables were nominal, ordinal or in an absolute scale. They selected a set of 11 projects for testing their models and the rest of the 63 was used for training the models. These two sets were selected as follows: the first training dataset was constructed by removing projects 1, 7, 13, 19, 25, 31, 37, 43, 49, 55, and 61; the second training dataset was constructed by removing projects 2, 8, 14, 20, 26, 32, 38, 44, 50, 56, 62, and so forth. The removed projects comprised the respective testing datasets. Results presented in this study indicated that these two techniques were competitive. The authors suggested that new experiments should be conducted to combine the neural network and multiple regression techniques to calibrate and test prediction models on other datasets, such as the ISBSG (International Software Benchmarking Standard Group) database.

Kultur et al. (2009) proposed an ensemble of neural networks with associative memory (ENNA). They used the following five datasets for evaluating the proposed model: the NASA dataset which includes 60 NASA projects from 1980 to 1990 (these projects are sequencing, avionics and mission planning projects), the NASA 93 dataset consisting of 93 NASA projects (these projects are from 1970 to 1980), the COCOMO 81 dataset containing 63 projects developed before 1981; the Softlab Data Repository (SDR) from the University Software Engineering Research Laboratory in Turkey (the SDR consists of 24 projects that were implemented after the year 2000), and the Desharnais dataset consisting of 77 projects from the late 1980s. The prediction accuracy evaluation criterion was the MMRE. The size by dataset used for training the ENNA was 50 (NASA), 78 (NASA 93), 53 (COCOMO 81), 20 (SDR) and 64 (Desharnais), whereas the size by dataset used for testing the ENNA was 10 (NASA), 15 (NASA 93), 10 (COCOMO 81), 4 (SDR) and 13 (Desharnais). In the ENNA model an ensemble is used rather than a single MLP. In addition to the use of the ensemble approach in this model, an associative memory is used for bias correction. The bias of the ensemble for similar projects is calculated and added to the result of the ensemble to provide the final result. The ENNA was compared with a multilayer perceptron neural network (MLP) and regression trees. Results showed that the ENNA produces better results than a MLP in terms of accuracy for all five datasets.

El-Sebakhy (2011) proposed functional networks as a new intelligent system paradigm and compared its performance with both standard multilayer perceptron and nonlinear statistical regression. The sample used was the same as the one used by Heiat (2002), and the prediction accuracy evaluation criterion was the MMRE. The results from this research indicated that the functional network learning scheme was competitive even better than both standard neural networks and multiple regression. El-Sebakhy suggested for future work to use more databases for software engineering.

As for the application of a general regression neural network (GRNN) on software engineering, Thwin and Quah (2005) used this kind of network for predicting software defects. The involved independent variables were object-oriented design metrics concerning inheritance, complexity, cohesion, and coupling measures, as well as memory allocation measures. In that study, the GRNN was found to predict more accurately than others models which the GRNN was compared with.

Regarding software development effort estimation, a GRNN was used by Prasad et al. (2010) for predicting large projects and by López-Martín (2011b) for predicting the effort of individual projects. Prasad et al. compared the accuracy of the GRNN with accuracies of the regression equation proposed by Boehm (1981) and that of a Radial Basis Neural Network (RBNN). These authors used the 63 projects from the COCOMO 81 dataset (developed before the year of 1981); 53 projects were used for generating the models and then all 63 projects were used for validating the generated models; that is, they used projects from the training set to validate their models. In addition, a GRNN spread value of 0.94 was used; however, the authors did not specify how that value was obtained. They concluded that the RBNN was better than the GRNN. On the other hand, López-Martín (2011b) used two sets of 163 and 80 individual projects for training and testing the GRNN, respectively; the independent variables were two kinds of lines of code, and the accuracy criteria were the Magnitude of error Relative to the Estimate (MER) and the Mean Square Error (MSE). Accuracy of the GRNN was compared with those of a linear regression and of a fuzzy logic model; results showed that a GRNN had a same statistically significant difference than the other two mentioned models.

2 Data Sample

In experiments where effort prediction models are compared, one concern has been "how repeatable the results are". This concern is because conventionally researchers have randomly partitioned the overall dataset into a training set and a validation set (Shepperd and Kadoda 2001). In this study, samples for training and for validating the models were not obtained randomly, but were selected based on a chronological manner.

A data sample of adjusted function points (AFP) corresponding to industrial projects was selected based on the suggested attributes presented in Table 6, taken from the "Selecting a Suitable Data Subset" section of the "Guidelines for use of the ISBSG data" document.

After application of the criteria mentioned in Table 6, the resulting sample had a size of 258 projects. The first 129 were selected for generating the models; however this sample was very asymmetric, with 31 of the projects presenting values from 520 to 17,518 AFP. Hence a subsample of 98 projects having from 6 to 499 AFPs was selected for generating the models. These 98 projects were developed from 1994 to 2001.

The "summary work effort" (hours) was converted to man-months applying the 152 h per man-month equivalence suggested by Boehm et al. (2000).

Table 6 Industrial data sample characteristics

Attribute	Selected value(s)
Data quality rating	A = The data submitted was assessed as being sound with nothing being identified that might affect its integrity.
	B = The submission appears fundamentally sound but there are some factors which could affect the integrity of the submitted data.
Unadjusted function point rating	A = The unadjusted function point count was assessed as being sound with nothing being identified that might affect its integrity.
	B = The unadjusted function point count appears sound, but integrity cannot be assured as a single figure was provided.
Development platform	Mainframe
Count approach	IFPUG
Functional sizing methods	IFPUG V4+
Resource level	1 = development team effort
Language type	3GL

3 Verification of Models

3.1 Regression Model

Using the data from the 98 projects, the following simple linear regression equation was generated:

$$\text{Effort} = 0.136577 + 0.0951844*\text{AFP}$$

The ANOVA p-value showed in Table 7 is 0.000; hence this equation had a statistically significant relationship between effort and AFP at the 95.0% confidence level. The plot of the fitted model is shown in Fig. 2. The value of its coefficient of determination was $r^2 \geq 0.5$, that is, 61% of the variation of the effort is explained by the model.

3.2 Applying the Models

The models presented in Sections 1.3 and 3.1 were applied to the original dataset and the MER by project as well as the MMER by model was then calculated.

The simple linear regression presented in section 3.1 yielded a MMER = 0.40.

In the GRNN, a parameter named SPREAD was changed until a suitable value was obtained. If the value of SPREAD is small, the GRNN function is very steep, so that the neuron with the weight vector closest to the input will have a much larger output than other neurons. The GRNN tends to respond with the target vector associated with the nearest

Table 7 ANOVA for simple linear regression of industrial projects

Source	Sum of squares	Degrees of freedom	Mean square	F-ratio	*p*-value
Model	12954.1	1	12954.1	150.30	0.0000
Residual	8273.89	96	86.1864		
Total	21228.0	97			

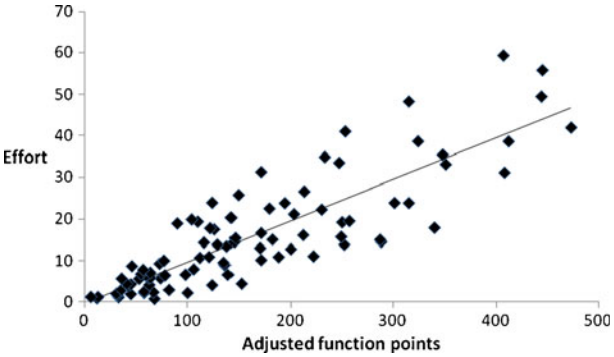


Fig. 2 Plot of the fitted model

input vector. As the SPREAD value becomes larger, the function slope of the GRNN becomes smoother and several neurons can respond to an input vector. The network then acts as if it is taking a weighted average between target vectors whose input vectors are closest to the new input vector. As the SPREAD value becomes larger, more and more neurons contribute to the average, with the result that the network function becomes smoother (Demuth et al. 2008).

The values for SPREAD were varied from 1 to 10, and then from 15 to 50 in 5-unit increments. The obtained MMER values are shown in Table 8.

4 Validation of Models

The models presented in Sections 1.3 and 3.1 were applied to a new dataset and the MER by project as well as the MMER by model was then calculated.

A new set of 129 projects obtained from the ISBSG database was considered for validating the models. These projects were developed from 2002 to 2008. Thirty-two projects were excluded from the set of 129 because they had values out of the size limits (minimum and maximum values) of the 98 projects used for generating the models. That is, a subsample integrated by 97 projects was used for validating the models.

The simple linear regression presented in section 3.1 yielded a MMER=0.45.

The values for the SPREAD parameter were again varied from 1 to 10, and then from 15 to 50 in 5-unit increments. The MMER values obtained are shown in Table 9, which contains the MMER values from Table 8 for comparison.

When selecting a suitable SPREAD value for the GRNN, a match between the MMERs from the verification and the validation stages is desired to have a balance between how well the resulting GRNN adjusts to the training dataset and how well it can predict the behavior of the validation dataset. With smaller SPREAD values, a better function fit is found for the training set of the verification stage, but the resulting GRNN might have a poor fit for the validation dataset (poor generalization). In general, the SPREAD value should be as small as possible without losing the ability to adequately predict the behavior of the validation set.

Table 8 MMER by SPREAD value in the verification stage

SPREAD value	1	2	3	4	5	6	7	8	9	10	15	20	25	30	35	40	45	50
MMER	0.16	0.24	0.29	0.31	0.33	0.34	0.35	0.35	0.36	0.36	0.38	0.39	0.40	0.40	0.40	0.41	0.41	0.41

Table 9 MMER by SPREAD value by stage

	SPREAD value																	
	1	2	3	4	5	6	7	8	9	10	15	20	25	30	35	40	45	50
Verification	0.16	0.24	0.29	0.31	0.33	0.34	0.35	0.35	0.36	0.36	0.38	0.39	0.40	0.40	0.40	0.41	0.41	0.41
Validation	0.67	0.58	0.53	0.49	0.49	0.49	0.48	0.48	0.47	0.46	0.44	0.43	0.42	0.42	0.42	0.42	0.41	0.41

Table 9 shows that when the SPREAD value in the verification and validation stages is equal to 45, the MMER has the best value for the validation stage (0.41); hence, predicted values of the trained model using this SPREAD value were used for comparing them against the ones generated by the regression model.

5 Comparison of Models

5.1 Verification Stage

The ANOVA p-value for MER of the projects (Table 10) shows that there was not a statistically significant difference in the accuracy of estimation for the two techniques at the 95.0% confidence level. The following three assumptions of residuals for MER ANOVA were then analyzed:

1) Independent samples: In this study, the software projects were separately made, so the data are independent.
2) Equal standard deviations: In a plot of this kind the residuals should fall roughly in a horizontal band centred and symmetrical about the horizontal axis (Fig. 3a), and
3) Normal populations: A normal probability plot of the residuals should be roughly linear (Fig. 3b).

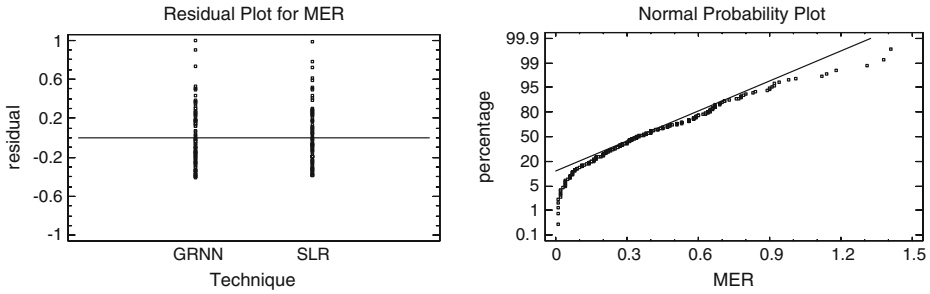
Since Fig. 3a and b suggest a slight abnormality, an additional Krustal-Wallis statistical test was done. The resulting p-value was equal to 0.7586, that is, there was not a statistically significant difference between the medians at the 95.0% confidence level, since its p-value was greater than 0.05.

5.2 Validation Stage

The ANOVA p-values for MER of the projects (Table 11) show that there was not a statistically significant difference in the estimation accuracy for the two techniques at the 95.0% confidence level, whereas the plots related to assumptions of residuals for MER ANOVA are shown in Fig. 4a and b.

Table 10 MER ANOVA (verification of industrial projects)

Source	Sum of squares	Degrees of freedom	Mean square	F-ratio	*p*-value
Between groups	0.00722	1	0.00722	0.08	0.7753
Within groups	17.1533	194	0.08841		
Total	17.1605	195			



(a) Equal standard deviations (b) Normality

Fig. 3 Plots of MER ANOVA (verification stage). **a** Equal standard deviations. **b** Normality

In Fig. 4a and b a slight abnormality can also be observed, hence an additional Krustal-Wallis statistical test was also done. The p-value obtained was equal to 0.7404, that is, there was not a statistically significant difference between the medians at the 95.0% confidence level.

6 Discussion

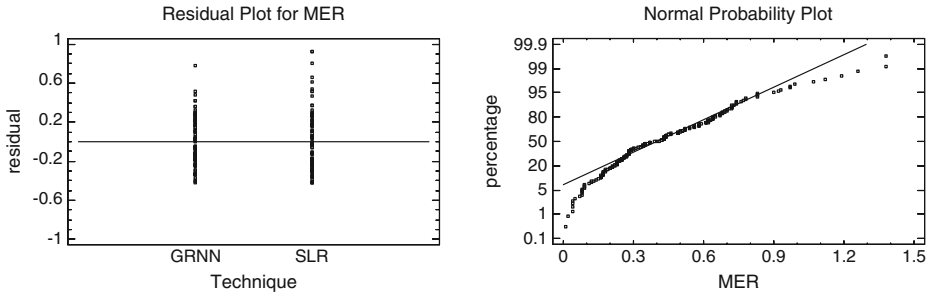
6.1 Neural Networks in Software Industry

As discussed in Section 1.4, many researchers have used NNs to produce models that estimate software development effort with an accuracy equal or better than other techniques, such as linear regression. However, despite the advantages of using neural networks to generate a model for predicting development effort, there has been reluctance from the part of practitioners to apply this kind of model to real-world projects (Park and Baek 2008). The main reasons NNs have weak support in effort estimation of industrial projects is their lack of explanation capabilities and the difficulty of finding the appropriate network architecture and parameter values that yield a useful model (Paliwal and Kumar 2009).

As for the first problem, since NNs are basically "black boxes" that take a number of inputs and produce one or more outputs, it is very difficult to interpret just how a given output can be derived from a given set of inputs. This difficulty arises from the fact that NNs usually work by applying a set of previously determined real-numbered "weights" to the input values along the various paths in the network and it is very hard to explain the NN behavior solely from these weight values. In order to study how to overcome the problem of poor interpretability of NNs when applied to effort

Table 11 MER ANOVA (validation stage of industrial projects)

Source	Sum of squares	Degrees of freedom	Mean square	F-ratio	p-value
Between groups	0.07680	1	0.07680	1.02	0.3132
Within groups	14.419	192	0.07509		
Total	14.4958	193			



(a) Equal standard deviations (b) Normality

Fig. 4 Plots of MER ANOVA (validation stage). **a** Equal standard deviations. **b** Normality

estimation, Idri et al. (2002) mapped a NN to a fuzzy rule-based system with the idea that if the obtained if-then fuzzy rules could be easily interpreted, then the NN could also be easily interpreted, given their equivalence. The NN they used was a three-layer perceptron with 13 input nodes, 13 hidden nodes and one output node. The 13 inputs corresponded to the same number of parameters taken from projects contained in the COCOMO 81 database, whereas the output corresponded to development effort. All projects from this database were used for training and testing the NN. Even though 13 fuzzy rules were obtained from the mapping of the NN, these authors had limited success in giving a straightforward interpretation of the rules obtained. They concluded that the operator they used to generate the fuzzy rules could not adequately model the complex set of relationships that existed between the input parameters. Furthermore, the method these authors used for mapping the three-layer NN to the fuzzy rule-based system cannot be applied to map a GRNN, which is the kind of NN that was used in the present work. More research remains to be done before the issue of interpretability of NNs is considered surmounted.

As for the second problem of using NNs for effort estimation of industrial projects, there are no guidelines for the construction of NNs regarding the number of layers, number of nodes per layer and initial weights (Idri et al. 2002). NNs models are usually built by trial and error, where a lot of experimenting has to be done to find a suitable combination of parameters and network topology. Even then, after training a NN to perform accurately for a given set of inputs, there is no guarantee that the NN will perform equally well on a different set of data. Practitioners tend to favor prediction models that can be easily recalibrated with data from other projects or environments. Another issue is the quality of the data used to train the NN, as it is well known that performance of a model based on NNs is highly dependent on the nature of the data itself, such as the level of noise and the appropriateness of the input variables. In many studies on effort estimation, a preliminary data analysis to remove outliers and non-dominant variables is not properly done. Consequently, practitioners may encounter difficulties when defining proper datasets from their real-world environments (Park and Baek 2008).

It should be mentioned that GRNNs present advantages over traditional NNs. The topology of a GRNN is in general not as variable as that of a NN, as the number of layers is usually predetermined in the former (Specht 1991). The training of GRNNs is also substantially faster than the methods used to train multi-layer perceptron networks, due to the use of an unsupervised method (which uses only the input data and not the

target data) to find the parameters of the hidden units, and a linear method to determine the final-layer weights (Bishop 1995).

6.2 Comparison with Previous Work

This study considered the four problems identified by Paliwal and Kumar (2009) that were described in section 1.4 ("Related work") of this study, which were approached as follows:

1) An ANOVA for the regression equation was done and its result showed a statistically significant relationship between dependent and independent variables at the 95.0% confidence level (Table 7).
2) An analysis of the SPREAD value for the GRNN was done in Section 4 and the results were presented in Table 9. These values were obtained using a software tool (MatLab 2009), which took only a few seconds to generate the mentioned MMER values.
3) We used separate samples for verifying and for validating the models. Projects for the two samples followed the criteria described in Table 6. The samples were chronologically separated based upon the year in which they were developed.
4) An ANOVA of accuracies of models (Tables 10 and 11) as well as assumptions of residuals were done (Figs. 3 and 4). Moreover, an additional Krustal-Wallis statistical test was also done (Sections 5.1 and 5.2) as the plots of residual analysis suggested slight abnormalities.

The projects in our study were developed from the year 1994 to the year 2008, whereas Heiat (2002) and El-Sebakhy (2011) based their conclusions on projects developed in 1988 or before; Oliveira (2006) used projects developed in 1981 or before, whereas Vinay et al. (2008) worked with data from projects that were developed in 1996 or before. Park and Baek (2008) used projects developed between 1999 and 2003; however, the sample they used was very heterogeneous. De Barcelos et al. (2008) used the COCOMO 81 dataset, which was published in 1981. Kultur et al. (2009) used five datasets developed respectively from 1980 to 1990, from 1970 to 1980, one dataset published in 1981, one developed after the year 2000, and one from the late 1980s; however, once their models were trained by the corresponding dataset, the dataset size used for testing their models was respectively 10, 15, 10, 4 and 13, whereas De Barcelos et al. used 11 projects for testing their models. Our study bases its conclusions on a sample of 97 more recent projects applied to the trained GRNN.

We followed the suggestions of El-Sebakhy (2011) and De Barcelos et al. (2008) in the sense that for future work, data from more recent projects should be used (De Barcelos et al. even suggested the use of the ISBSG repository).

In relation to GRNN, we can only compare our study with that of Prasad et al. (2010), as they trained a GRNN for predicting the effort of industrial projects. Prasad et al. (2010) used the projects developed before 1981: 53 of them were used for training the models and then the 63 projects were used for validating their models (i.e., they used the same sample for training and validating their models). Moreover, Prasad et al. (2010) did not specify how the spread value for the GRNN was obtained.

The projects in this study only involved projects coded with third generation programming languages, whereas Heiat (2002) combined languages from two generations. All studies based on industrial projects described in Section 1.4 ("Related work") of this study used the Mean Magnitude of Relative Error (MMRE) as accuracy criteria, whereas we used the Mean Magnitude of error Relative to the Estimate (MMER) as it showed better results than MMRE in Foss et al. (2003).

7 Conclusions and Future Research

This study compared two models for estimating and predicting development effort for projects developed in industrial environments. One of the models studied—statistical regression—is one of the most used in the software estimation field, whereas the other corresponded to a computational intelligence technique: a neural network. Data samples were obtained from the ISBSG repository. For the verification and validation stages of the models, 98 and 97 projects were used, respectively.

The accepted hypothesis was the following: effort prediction accuracy of a general regression neural network is statistically equal than that obtained by a multiple linear regression, when data is obtained from industrial environments.

These results suggest that a GRNN represents an alternative for predicting development effort of projects, measured in function points and developed in industrial environments.

Future research involves the use of bidirectional associative memories for predicting the development effort of software projects.

Acknowledgment Authors of this paper would like to thank CUCEA of Guadalajara University, Jalisco, México, Programa de Mejoramiento del Profesorado (PROMEP), as well as to Consejo Nacional de Ciencia y Tecnología (Conacyt).

References

Afzal W, Torkar R (2011) On the application of genetic programming for software engineering predictive modeling: a systematic review. J Expert Syst Appl, Elsevier 38:11984–11997. doi:10.1016/j.eswa.2011.03.041

Bishop Ch M (1995) Neural networks for pattern recognition, Oxford

Boehm B (1981) Software engineering economics, Prentice Hall

Boehm B, Abts Ch, Brown AW, Chulani S, Clarck BK, Horowitz E, Madachy R, Reifer D, Steece B (2000) COCOMO II. Prentice Hall

De Barcelos TIF, Simies da Silva JD, Sant Anna N (2008) An investigation of artificial neural networks based prediction systems in software project management. J Syst Software, Elsevier 81:356–367. doi:10.1016/j.jss.2007.05.011

Demuth H, Beale M, Hagan M (2008) MatLab neural network toolbox 6, User's Guide

Duda R, Hart P, Stork D (2001) Pattern classification. Second Edition. Willey-Interscience

El-Sebakhy EA (2011) Functional networks as a novel data mining paradigm in forecasting software development efforts. J Expert Syst Appl, Elsevier 38:2187–2194. doi:10.1016/j.eswa.2010.08.005

Foss T, Stensrud E, Kitchenham B, Myrtveit I (2003) A simulation study of the model evaluation criterion MMRE. IEEE Trans Softw Eng 29(11):985–995. doi:10.1109/TSE.2003.1245300

Garmus D, Herron D (1996) Measuring the software process, a practical guide to functional measurements. Prentice Hall

Haykin S (1999) Neural networks, a comprehensive foundation, Second Edition, Pearson

Heiat A (2002) Comparison of artificial neural network and regression models for estimating software development effort. J Inform Software Tech, Elsevier 44(15):911–922. doi:10.1016/S0950-5849(02)00128-3

Idri A, Khoshgoftaar TM, Abran A (2002) Can neural networks be easily interpreted in software cost estimation? IEEE Int Conf Fuzzy Syst: 1162–1167. doi:10.1109/FUZZ.2002.1006668

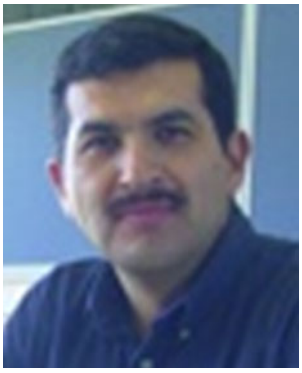
Jørgensen M (2006) A preliminary theory of judgment-based project software effort predictions. In: Ou L, Turner R (eds) IRNOP VIII. Project research conference. Publishing House of Electronic Industry, Beijing, pages 661–668

Jørgensen M (2007a) A critique of how we measure and interpret the accuracy of software development effort estimation. 1st International Workshop on Software Productivity Analysis and Cost Estimation, pages 15–22

Jørgensen M (2007b) Forecasting of software development work effort: evidence on expert judgment and formal models. J Forecast, Elsevier 23(3):449–462. doi:10.1016/j.ijforecast.2007.05.008

Jørgensen M, Shepperd MJ (2007) A systematic review of software development cost estimation studies. IEEE Trans Softw Eng 33(1):33–53

Kitchenham BA, Mendes E, Travassos GH (2007) Cross versus within-company cost estimation studies: a systematic review. IEEE Trans Softw Eng 33(5):316–329
Kultur Y, Turhan B, Bener A (2009) Ensemble of neural networks with associative memory (ENNA) for estimating software development costs. J Knowl Base Syst, Elsevier 22:395–402. doi:10.1016/j.knosys.2009.05.001
López-Martín C (2011a) A fuzzy logic model for predicting the development effort of short scale programs based upon two independent variables. J Appl Soft Comput, Elsevier 11(1):724–732. doi:10.1016/j.asoc.2009.12.034
López-Martín C (2011b) Applying a general regression neural network for predicting development effort of short-scale programs. J Neural Comput Appl, Springer 20:389–401. doi:10.1007/s00521-010-0405-5
MatLab (2009) Version 7.8.0.347 (R2009a), MathWorks, Inc.
Montgomery D, Peck E (2001) Introduction to linear regression analysis. John Wiley
Myrtveit I, Stensrud E, Shepperd M (2005) Reliability and validity in comparative studies of software prediction models. IEEE Trans Softw Eng 31(5):380–391. doi:10.1109/TSE.2005.58
Oliveira ALI (2006) Estimation of software project effort with support vector regression. Neurocomputing, Elsevier 69:1749–1753. doi:10.1016/j.neucom.2005.12.119
Paliwal M, Kumar UA (2009) Neural networks and statistical techniques: a review of applications. J Expert Syst Appl 36:2–17. doi:10.1016/j.eswa.2007.10.005
Park S, Baek S (2008) An empirical validation of a neural network model for software effort estimation. J Expert Syst Appl, Elsevier 35:929–937. doi:10.1016/j.eswa.2007.08.001
Prasad Reddy PVGD, Sudha KR, Rama SP, Ramesh SNSVSC (2010) Software effort estimation using radial basis and generalized regression neural networks. J Comput 2(5), May 2010
Sheetz SD, Henderson D, Wallace L (2009) Understanding developer and manager perceptions of function points and source lines of code. J Syst Software, Elsevier 82:1540–1549. doi:10.1016/j.jss.2009.04.038
Shepperd M, Kadoda G (2001) Comparing software prediction techniques using simulation. IEEE Trans Softw Eng 27(11):1014–1022. doi:10.1109/32.965341
Specht DF (1991) A general regression neural network. IEEE Trans Neural Netw 7(3):568–576. doi:10.1109/72.97934
Sun-Jen H, Nan-Hsing Ch, Li-Wei Ch (2008) Integration of the grey relational analysis with genetic algorithm for software effort estimation. J Oper Res, Elsevier 18:898–909. doi:10.1016/j.ejor.2007.07.002
Thwin MMT, Quah TS (2005) Application of neural networks for software quality prediction using object-oriented metrics. J Syst Software, Elsevier 2:147–156. doi:doi:10.1016/j.jss.2004.05.001
Vinay KK, Ravi V, Carr M, Raj KN (2008) Software development cost estimation using wavelet neural networks. J Syst Software, Elsevier 81:1853–1867. doi:10.1016/j.jss.2007.12.793



Cuauhtémoc López-Martín is researcher with the Information Systems Department at the Universidad de Guadalajara, Jalisco, Mexico. He received his Ph. D. in Computer Science in the Center for Computing Research of the National Polytechnic Institute of Mexico (2007). His research is related to software estimation techniques, software processes and Statistics applied to software engineering. He is member of the Mexican National System of Researchers. He has been reviewer of journals as Fuzzy Sets and Systems (Elsevier), the Journal of Systems and Software (Elsevier), International Journal of Software Engineering and Knowledge Engineering (IJSEKE), and of the Central European Journal of Computer Science.

Claudia Isaza is an Assistant Professor in Electronic Engineering at the University of Antioquia, Medellín, Colombia. Her research interests include complex systems monitoring using clustering methods, data mining, fuzzy logic. She received her bachelor´s degree in electronic engineering from Distrital F.J.C. University (Bogota-Colombia) in 2002, her M.S. degree in electrical engineering (control emphasis) from Andes University (Bogota-Colombia) in 2004, and her Ph.D. degree in Automatic Systems from INSA, Toulouse, France in 2007.

Arturo Chavoya is a professor with the Information Systems Department at the Universidad de Guadalajara, Mexico working on subjects related to Artificial Intelligence, Artificial Life, Parallel Programing and Bioinformatics. Professor Chavoya received his Ph.D. in Computer Science from the Université de Toulouse in France. He received his M.S. in Computer Science from the Centro de Investigación y de Estudios Avanzados del I.P.N (CINVESTAV) Unidad Guadalajara, Mexico, his B.S. in Computer Systems Engineering from ITESO University in Guadalajara, Mexico, and his B.S. in Biology from the University of Guadalajara in Mexico.