RBFN Networks-based Models for Estimating Software Development Effort: A Cross-validation Study

Ali Idri, Aya Hassani
Department of Software Engineering
ENSIAS, Mohammed V University
Rabat, Morocco
idri.ali123@gmail.com, hassani.aya@gmail.com

Alain Abran
Department of Software Engineering
Ecole de Technologie Supérieure
Montréal, Canada, H3C IK3
alain.abran@etsmtl.ca

***Abstract*— Software effort estimation is very crucial and there is always a need to improve its accuracy as much as possible. Several estimation techniques have been developed in this regard and it is difficult to determine which model gives more accurate estimation on which dataset. Among all proposed methods, the Radial Basis Function Neural (RBFN) networks models have presented promising results in software effort estimation. The main objective of this research is to evaluate the RBFN networks construction based on both hard and fuzzy C-means clustering algorithms using cross-validation approach. The objective of this replication study is to investigate if the RBFN-based models learned from the training data are able to estimate accurately the efforts of yet unseen data. This evaluation uses two historical datasets, namely COCOMO81 and ISBSG R8.**

I. INTRODUCTION

Software development effort estimation is very important for the successful completion of any software project. Hence, software development effort and schedule can be predicted on the basis of past software project datasets. Effort estimation of software projects can be done basically in three ways [1,2,3]:

Judgment based on experts: That aims to derive estimates based on experience of experts on similar projects. It is based on a tacit (intuition-based) quantification step [4]. Therefore, it is not repeatable. According to Gray et al. [5] although expert judgment is always difficult to quantify, it can be an effective tool to adjust both machine learning (ML) and non-ML techniques.

Non-machine learning (Non-ML) techniques: They are based on mathematical formulae linking effort with effort drivers to produce an estimate of the project [6,7]. Usually the principal effort driver used in these models is software size measured in terms of function points or source lines of code.

Machine learning (ML) techniques: Effort estimation is based on applying various machine learning algorithms such as artificial neural networks (ANN) [8,9], case-based reasoning (CBR)[10,11,12], decision trees [13,14] and fuzzy logic [15,16]. These models have received recently increasing attention from researchers [2,3].

ANNs have been investigated in software effort estimations because they can interpret the complex relationships among software project features. Moreover they have learning ability and are good at modeling complex non-linear relationships but they are still hard to understand and/or to interpret [3,17,18,] However, the use of ANNs in software effort estimation is far from mature. Among all types of ANNs, feedforward ANNs have been widely used for software effort estimations [19-23]. Usually the number of layers, number of neurons in each layer, selection of activation function, quality of data, over-fitting, and outliers are the main subjects to build an ANN-based software effort estimation model [3].

In our earlier works [24-26], three clustering techniques (i.e., APCIII, hard C-means and fuzzy C-means, were empirically evaluated to design the middle layer of RBFN for software effort estimation. It has been illustrated that: 1) the RBFN designed with C-means performs better, in term of effort estimates accuracy, than the RBFN designed with APCIII algorithm; and 2) an RBFN using fuzzy C-means performs better, in terms of accuracy and tolerance of imprecision, than an RBFN using hard C-means. However, the importance of these findings is limited since the accuracy of any estimation technique (for instance RBFN), is highly depended on the characteristics of the used dataset, especially its sample size. Consequently, that study [27] used the International Software Benchmarking Standards Group repository-ISBSG release 8 dataset [28] which contains more than 2000 historical software projects from which 151 software projects were retained to evaluate the estimation accuracy of RBFN-based effort estimation models. The results showed that the accuracy of RBFN construction-based on fuzzy C-means generates more accurate estimates than an RBFN using hard C-means especially when decreasing the number of clusters. Moreover, we have found that an RBFN using fuzzy C-means better tolerates imprecision than an RBFN using hard C-means and hence may avoid the over-fitting problem. Since all our

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.142

previous studies on RBFN-based effort estimation used the same data for learning and testing stages of the model, this paper revisits them by means of a specific cross-validation approach on two different datasets. Therefore, this study evaluates the estimates accuracy of the RBFN-based models on both learning and testing sets in order to analyze their strengths and weaknesses in terms of accuracy, robustness and generalization.

The rest of the paper is organized as follows: Section 2 presents how an RBFN network may be used for software effort estimation. Section 3 describes the two datasets and the accuracy performance measures used in this study. Section 4 describes our approach to estimate required software effort with the experimental setup i.e., datasets preparation and implementation details using cross-validation approach. Section 5 presents the results obtained when applying different RBFN-based effort estimation models on the datasets. Finally, Section 6 summarizes our findings.

II. RBFN Network for Software Effort Estimation

Radial Basis Function Networks are a special case of artificial neural networks, rooted in the idea of biological receptive fields [29]. Figure 1 shows a RBFN architecture configured for software development effort. An RBFN is a three-layer feed forward network consisting of one input layer, one middle layer and an output layer. The RBFN generates output (effort) by propagating the initial inputs (cost drivers) through the middle-layer to the final output layer. Each input neuron corresponds to a component of an input vector. The input-layer contains M neurons, each input neuron is fully connected to the middle-layer neurons. The activation function of each middle neuron is usually the Gaussian function. The Gaussian function decreases rapidly if the width σi is small, and slowly if it is large. The output layer consists of one output neuron that computes the software development effort as a linear weighted sum of the outputs of the middle layer. Clustering is the key technique to be used as a preprocessing phase in the design of the RBFN networks. This procedure initially distributes the respective fields of hidden layer neurons across the space of input variables.

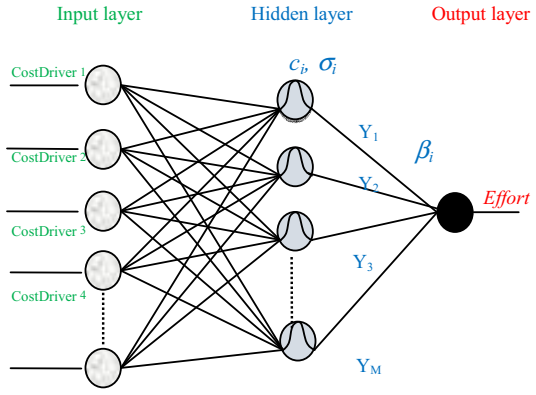

Fig. 1. Radial basis function network.

III. Datasets Description and Performance Evaluation Criteria

This section describes the datasets used in this study and the evaluation criteria used to compare the predictive accuracy of the designed models.

A. Datasets Description

Two datasets are used in this empirical study. Table 1 provides some statistics on these datasets including the total number of projects, the number of attributes taken into consideration, minimum, maximum and the mean of real efforts.

TABLE I. Statistics on Data Sets.

Dataset	**#Projects**	**#Attributes**	**Real effort** (Person-Months for COCOMO81 and Person-Hours for ISBSG)		
			Min	***Max***	***Mean***
ISBSG	151	6	24	60270	5039.13
COCOMO81	252	13	6	11400	683.44

ISBSG repository. The International Software Benchmarking Standards Group is a multi-organizational dataset [28] containing different projects gathered from different organizations in different countries. Major contributors come from Australia (21%), Japan (20%) and United States (18%). ISBSG repository release 8 contains more than 2000 software projects from which 151 software projects were retained. The selection of software projects bas been performed by choosing the software projects with high quality data according to the following four criteria: Data Quality Rating field, Resource levels, Unadjusted Function Points (UFP) and Development Type [27]. The selection of the attributes to be used as the inputs (effort drivers) of an effort estimation model based on an RBFN network has been conducted by allowing the estimators to use the attributes that they believe best characterize their projects and are more appropriate in their environment. The projects in ISBSG dataset are described by more than 50 attributes. Moreover, the use of hard or fuzzy C-means requires numerical attributes. Consequently, the selected attributes must be numerical. Table 2 shows the six attributes that have been selected since they were usually considered in the previous studies [27] as relevant cost drivers.

TABLE II. Software Project Attributes for ISBSG Dataset

Attributes	**Description**
Value Adjustment Factor/ VAF	The adjustment of the function points
Unadjusted Function Points/ UFP	The unadjusted function point count
Max Team Size	The maximum number of people that worked at any time on the project
User Base/ Business Units	Number of business units that the system services

Attributes	Description
User Base/ Locations	Number of physical locations being serviced by the installed system.
User Base/ Concurrent Users	Number of users using the system concurrently.
Value Adjustment Factor/ VAF	The adjustment of the function points

Artificial COCOMO81 dataset. The original COCOMO81 dataset contains 63 projects which are mostly scientific applications developed by Fortran [6]. However, it is suitable that the use of ANNs in any field should be based on a sufficient number of cases. Therefore, we used in this study an artificial COCOMO81 dataset generated from the original one [30]. It contains 252 software projects which are mostly scientific applications developed by Fortran generated from the original one [6]. As shown in Table 3, each project is described by 13 attributes. The software size measured in KDSI (Kilo Delivered Source Instructions) and the remaining 12 attributes are measured on a scale composed of six linguistic values: 'very low', 'low', 'nominal', 'high', 'very high' and 'extra high'. These 12 attributes are related to the software development environment such as the experience of the personnel involved in the project, the method used in the development and the time and storage constraints imposed on the software.

TABLE III. SOFTWARE PROJECT ATTRIBUTES FOR COCOMO81 DATASET.

Attributes	Description
SIZE	Software size
DATA	Database size
VIRTMIN	Virtual machine volatility
TIME	Execution time constraints
STOR	Main storage constraints
VIRT MAJ	Virtual machine volatility
TURN	Computer turnaround
ACAP	Analyst capability
AEXP	Application experience
PCAP	Programmer capability
VEXP	Virtual machine experience
LEXP	Programming language and tool experience
SCED	Required development Schedule

B. Data Normalization

Due to the nature of the attributes in the two datasets, some of continuous attributes show a larger range of values than others which may make the effect of these attributes too important. In order to normalize the two datasets used, the solution was to scale the continuous attributes into the same range. To achieve this, all continuous attributes are normalized applying the min-max normalization formula of Equation (1) such that all numerical variables are scaled within the range of [0, 1].

$$x_i(k) = \frac{x_i(k) - \min\,(x(k))}{\max\big(x(k)\big) - \min\,(x(k))} \quad (1)$$

Where $\min\big(x(k)\big)$is the minimum value of the dataset $x(k)$, $\max\big(x(k)\big)$is the maximum value of the dataset $x(k)$, if $\max\big(x(k)\big)$is equal to the $\min\big(x(k)\big)$, then Normalized $x_i(k)$ is set to 0.5.

C. Performance measures

The accuracy of the model is evaluated by using the following criteria:

The Magnitude of Relative Error. MRE is a very common criterion used to evaluate software cost estimation models. The MRE for each observation obtained as:

$$MRE = \left|\frac{Effort_{actual} - Effort_{predicted}}{Effort_{actual}}\right| \quad (2)$$

The Mean Magnitude of Relative Error. MMRE can be achieved through the summation of MRE over N observations.

$$MMRE = \frac{\sum_{i=1}^{N} MRE}{N} \quad (3)$$

The Percentage of the Prediction. PRED is computed as follow:

$$Pred\ (X) = \frac{A}{N} \quad (4)$$

Where, A is the number of projects with MRE less than or equal to X, and N is the total number of projects. Usually the ideal amount of X in software effort estimation methods is 0.25 and the various methods are compared based on this level. The prediction at 25%, Pred(25), represents the percentage of projects which MRE is less or equal to 25%. Pred(25) value identifies the software effort estimates that are within the specified range whereas the MMRE is fairly conservative with a bias against overestimates. Pred(25) can take a value between 0 and 100 percent, while the MMRE can take any positive value. It is often difficult to compare results across different studies due to differences in empirical setup and data preprocessing, but a typical Pred(25) lies in the range of 65 to 100 percent.

IV. EMPIRICAL DESIGN

The potential over-fitting problem of a neural network should be dealt by applying methods such as cross-validation in order to evaluate the accuracy and evaluate the generalization level of the models. Cross-validation methodology is used for comparing models by dividing data into two segments: one used to learn or train a model and the other used for testing to validate the model. In typical cross-validation, the training set and testing set must cross-over in successive rounds such that each data point has a

chance of being validated against. The basic form of cross-validation is k-fold cross-validation; other used forms are special cases of k-fold cross-validation or involve repeated rounds of k-fold cross-validation. In k-fold cross-validation, the data is first partitioned into k equally (or nearly equally) sized segments or folds. Subsequently k iterations of training and testing are performed.

In this study, a cross-validation technique is used on the one hand, to evaluate the generalization performance of the trained RBFN model construction-based either on hard or fuzzy C-means and, on the other hand, to compare the performance of the designed RBFN models to find out the best model for the available data. More specifically, the cross-validation technique used in this work for evaluating the performance of RBFN consists on having different data in training and testing. Indeed, 10 equal size collections are obtained for either training or testing sets by partitioning the original datasets; this technique will allow us to make the best use of our (limited) datasets for training, testing and performance evaluation.

To decide on the number of hidden units, several tests have been performed with both hard C-means and fuzzy C-means algorithms. Choosing the best classification to determine the number of hidden neurons and their centers is not an obvious task. For software effort estimation, the best classification is the one that provides coherent clusters which have satisfactory degrees of similarity, and improve the accuracy of estimates, in terms of MMRE and Pred(25), produced by the RBFN. Indeed, the execution of our software prototypes which implement hard and fuzzy C-means clustering algorithms together with the software prototype implementing the RBFN model are repeated 10 times. The means of obtained results from 10 iterations are treated as the final result of MMRE and Pred(25). Figure 2 shows the evaluation process of an RFBN-based effort estimation model using the cross-validation strategy.

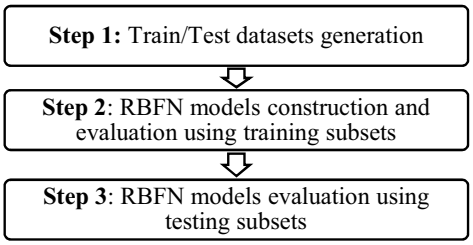

Fig. 2. Steps of the evaluation process

A. Step 1: Train/Test datasets generation.

This step consists in dividing each dataset on two subsets to be used for training and testing the RBFN effort estimation models respectively. Table 4 shows the number of projects of each dataset for training and testing. This training/testing subsets generation process was repeated 10 times for each dataset with the same percent of Table 4. Hence, for each dataset, 10 different collections of training/testing subsets have been obtained. Hence, for each dataset, 10 different collections of training/testing subsets have been obtained.

TABLE IV. TRAINING AND TESTING SETS GENERATION

Datasets	#Training Projects	%Training Set	#Testing Projects	%Testing Set
ISBSG	100	66.23%	51	33.77%
COCOMO81	200	79.37%	52	20.63%

B. Step 2: RBFN models construction and evaluation using training subsets.

This step consists on the construction and evaluation of RBFN-based effort estimation models using the training subsets of the 10 collections of each dataset. The use of an RBFN on each training subset of a collection to estimate software development effort requires the determination of its architecture parameters according to the characteristics of the training software projects, especially the number of input neurons, number of hidden neurons, centers cj, widths j and weights βj. The number of the input neurons is, usually, the number of the attributes describing the training software projects of the used dataset. Therefore, according to Table 1, the number of input neurons of the RBFN models of ISBSG and COCOMO81 is 6 and 13 respectively.

The construction of the hidden layer of the proposed RBFN networks is achieved using hard and fuzzy C-means. The role of clustering in the design of RBFN is to set up an initial distribution of receptive fields (hidden neurons) across the input space of the input variables (effort drivers). For the widths σj, many techniques based on solid mathematical concepts have been proposed in the literature [31, 32, 33]. Based on our previous results [24-27] in which accurate estimates were obtained when using the formula defined by Haykin [34], we adopt this solution which consists in assigning one value to all (σj). Concerning the weights βj, we may set each βj to the associated effort of the center of the j^{th} neuron. But this technique is not optimal and does not take into account the overlapping that may exist between receptive fields of the hidden layer. Thus, we use the learning Delta rule to derive the values of βj.

For each collection of a dataset and based on its training subset, 10 evaluations have been performed with both hard and fuzzy C-means algorithms to decide on the number of hidden units of an RBFN model. Thereafter, for each collection/training subset, the best classification is the one that provides coherent clusters which have satisfactory degrees of similarity and improve the accuracy of estimates, in terms of Pred(25). Hence the chosen RBFN models of each collection/training subset are those with acceptable Pred(25) (around 65%) and have minimum number of hidden neurons (clusters). To summarize, in this step we have performed in total 400 (2datasets* 10collections* 10evaluations* 2clustering techniques) evaluations for RBFN-construction based on hard and fuzzy C-means.

C. Step 3: RBFN models evaluation using testing subsets.

This step consists on the evaluation of the 'best' RBFN models of the step 2 using the testing subsets of the 10 collections of each dataset. The aim is to evaluate the generalization power of the designed RBFN models. The total number of evaluations X of each dataset is calculated with the following equation:

$$X = 2 \times \sum_{i=1}^{10} \#best\ RBFN(collection/training)i \qquad (5)$$

Where #best RBFN(collection/training) is the number of selected RBFN models which have Pred(25) around 65% and minimize the numbers of clusters during training for each triple (dataset, collection, training set). Thereafter, each 'best' RBFN selected model is executed 10 times on the testing subsets; the mean value of Pred(25) is calculated to analyze the results of testing phase.

V. Results and Discussion

A. Cross-validation of RBFN-based on hard C-means

The To measure the coherence of clusters in case of hard C-means, the objective function J, rather than the Dunn's validity index, is used. Indeed, in our previous empirical studies [26-27], the use of J had led to accurate estimates when applying an RBFN based on C-means to different datasets. The evaluations of the RBFN construction-based on C-means were performed according to the process of Figure 2.The obtained results are presented and discussed below.

For each collection of the ISBSG or COCOMO81 datasets, we have performed several evaluations, 1) 10 evaluations on its training subsets to build the RBFN models and 2) several evaluations on its testing subsets to test the 'best' constructed RBFN models. For each chosen 'best' RBFN model, 10 evaluations on the testing subsets have been conducted. Since the number of 'best' constructed RBFN models may change from one dataset to another, we have chosen: 1) For the ISBSG dataset, 13 'best' RBFN models with number of hidden units (clusters) within the interval [76, 90]; for each one, 10 evaluations were performed on the testing subsets. Hence, the total number of evaluations for the ISBSG dataset is 13000=10collections*10training-subset*13'best' RBFN models*10testing-subsets. 2) For the COCOMO81 dataset, 16 'best' RBFN models with number of hidden units within the interval [150, 180]. Hence, the total number of evaluations for the COCOMO81 dataset is 16000=10collections*10training-subset*16'best'RBFN-models * 10 testing-subsets.

Figures 3-4 show and compare the accuracy measured in terms of Pred(25) of an RBFN construction-based on hard C-means using the ISBSG and COCOMO81 datasets respectively. The mean of Preds(25) is used to represent the values of Pred(25) obtained when evaluating an RBFN construction-based on hard C-means using the 10 ISBSG/COCOMO81 collections. We can notice that the accuracy in terms of the Pred(25) criterion is, in general, monotonous increasing according to c, number of clusters (hidden neurons), for both training and testing evaluations. This is normal because when the number of clusters is higher (tends towards 90 and 180 for ISBSG and COCOMO81 respectively) the clusters obtained are more coherent; for example if c is equal to 90 for ISBSG, each cluster contains one project. However, the aim is to reduce the number of clusters c and to keep the estimates accuracy acceptable. Consequently, we are concerned with the accuracy when the values of c tend towards zero.

As shown in Figures 3-4, the accuracy of an RBFN construction-based on hard C-means on the training (learning) subsets is in general higher than that one obtained on testing subsets since building the RBFN model used training subsets. To this regard, it is noticed that: 1) For the ISBSG dataset, when c is lower than 77, the RBFN estimates accuracy in terms of Pred(25) is not acceptable (Pred(25)<60%) in training subsets; it is the same in testing subsets when c is lower than 88. 2) For the COCOMO81 dataset, when c is lower than 158, the RBFN estimates accuracy in terms of Pred(25) is not acceptable (Pred(25)<60%) in training subsets; it is the same in testing subsets when c is lower than 160. If we consider the accuracy of an RBFN on training and testing, we conclude that the threshold value of c to use an RBFN based on hard C-means is around 88 and 160 for the ISBSG and COCOMO81 datasets respectively. This is very high considering that the maximum possible value is 90 (ISBSG) and 180 (COCOMO81) clusters. It may be due to the fact that software projects are not sufficiently similar.

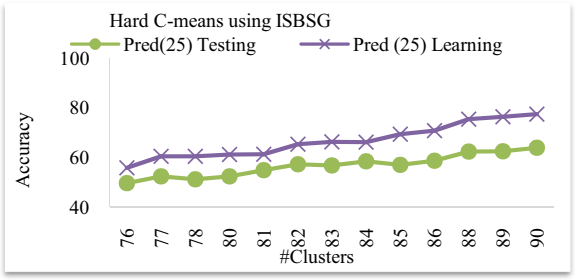

Fig. 3. Relationship between the Pred(25) of an RBFN and the number of clusters generated by the hard C-means for the ISBSG dataset.

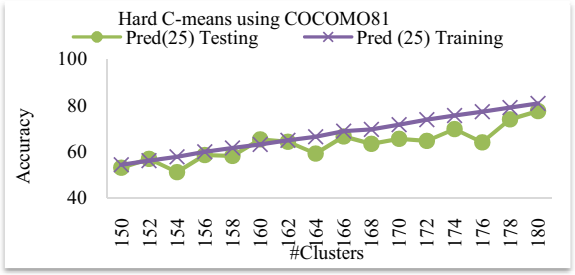

Fig. 4. Relationship between the Pred(25) of an RBFN and the number of clusters generated by the hard C- means for the COCOMO81 dataset.

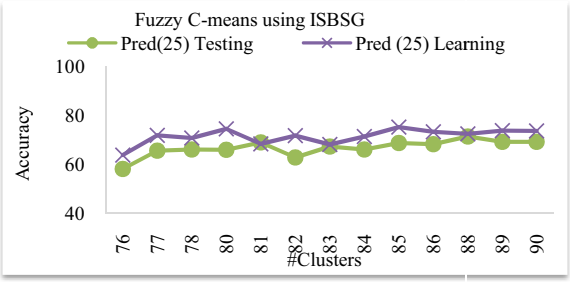

Fig. 5. Relationship between the Pred(25) of an RBFN and the number of clusters generated by the fuzzy C-means for the ISBSG dataset.

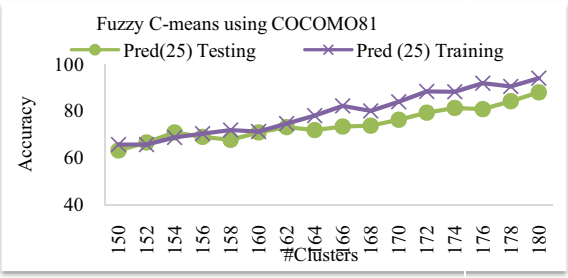

Fig. 6. Relationship between the Pred(25) of an RBFN and the number of clusters generated by the fuzzy C-means for the COCOMO81 dataset.

B. Cross-validation of RBFN-based on Fuzzy C-means

To measure the coherence of clusters in case of fuzzy C-means, the Xie-Beni validity criterion is used [38]. A small value of XB means a more compact and separate clustering. The goal should therefore be to minimize the value of XB and keep the estimates accuracy acceptable. Similarly to the case of hard C-means, for each dataset/collection, 10 evaluations were performed on its training subsets to build the RBFN models. Regarding evaluations on its testing subsets, we choose the same number of 'best' RBFN construction-based on fuzzy C-means as in the case of hard C-means, namely 13 (ISBSG) and 16 (COCOMO81). For each 'best' RBFN model, 10 evaluations on each testing subset have been conducted.

Figures 5-6 show and compare the accuracy measured in terms of Pred(25) of an RBFN construction-based on fuzzy C-means using the ISBSG and COCOMO81 respectively. As in the case of hard C-means, the mean of Preds(25) is used to represent the values of Pred(25) obtained when evaluating an RBFN construction-based on fuzzy C-means using the 10 ISBSG/COCOMO81 collections.

Figure 5-6 illustrate the relationship between the accuracy of an RBFN architecture and the number of clusters generated by the fuzzy C-means for both training and testing subsets of the 10 ISBSG/COCOMO81 collections. As in the case of hard C-means, we can notice that: 1) the accuracy in terms of the Pred(25) criterion is, in general, monotonous increasing according to c, number of clusters (hidden neurons), for both training and testing evaluations; and 2) the accuracy of an RBFN construction-based on fuzzy C-means on the training subsets is in general higher than that one obtained on testing subsets since building. To this regard, it is noticed that: 1) For the ISBSG dataset, when c is lower than 76, the RBFN estimates accuracy in terms of Pred(25) is not acceptable (Pred(25)<60%) in training subsets; it is the same in testing subsets when c is lower than 77. 2) For the COCOMO81 dataset, when c is lower than 158, the RBFN estimates accuracy in terms of Pred(25) is not acceptable (Pred(25)<60%) in training subsets; it is the same in testing subsets when c is lower than 160.

If we consider the accuracy of an RBFN on training and testing, we conclude that the threshold value of c to use an RBFN based on fuzzy C-means is around 77 and 150 for the ISBSG and COCOMO81 datasets respectively..

C. RBFN using Hard C-means vs RBFN using Fuzzy C-means

In this section, we compare in terms of accuracy and tolerance of imprecision an RBFN based on fuzzy C-means with the one when using hard C-means. The comparison is based on both training and testing sets.

Figure 7-8 compare the Pred(25) of an RBFN using the fuzzy c-means with that one of an RBFN when using the hard C-means on training subsets of ISBSG and COCOMO81 respectively. It can be noticed that when increasing number of clusters (the maximum is the number of training projects), hard C-means performs better than fuzzy C-means due to the fact that each cluster tend to contain only one project and since this project is used in clustering and training the RBFN, hard C-means classify each project in its associate cluster; hence the estimate effort is accurate. For the ISBSG dataset, when c is higher than 86 and tends towards 90, the accuracy of an RBFN based on C-means is better than that one of an RBFN based on fuzzy C-means. Additionally, For the COCOMO81 dataset, whatever the number of clusters the accuracy of an RBFN based on fuzzy C-means is better than that one of an RBFN based on hard C-means.

As the aim is to reduce the number of clusters with keeping a better accuracy, it is very important to consider the accuracy when decreasing the number of clusters (c tends towards zero). From this point of view, an RBFN based on fuzzy C-means performs better than an RBFN based on hard C-means for the two datasets. This is due to the fact that when decreasing the number of clusters, hard C-means is forced to classify each project in one cluster while fuzzy C-means may allow a project to belong to more than one cluster. Hence the possibility of miss-classification of hard C-means is higher than that of fuzzy C-means.

Figures 9-10 compare the Pred(25) of an RBFN using the fuzzy c-means with that one of an RBFN when using the hard C-means on testing subsets of ISBSG, and COCOMO8 respectively. It can be noticed that an RBFN based on fuzzy C-means performs better that an RBFN based on hard C-means for the two datasets. To this regard, it is noticed that:

1) For the ISBSG dataset, the accuracy of an RBFN based on fuzzy C-means is still achieved until c is equal to 77 while it is only to 88 when using an RBFN based on hard C-means.

2) For the COCOMO81 dataset, the accuracy of an RBFN based on fuzzy C-means is still achieved until c is equal to 150 while it is only to 160 when using an RBFN based on hard C-means.

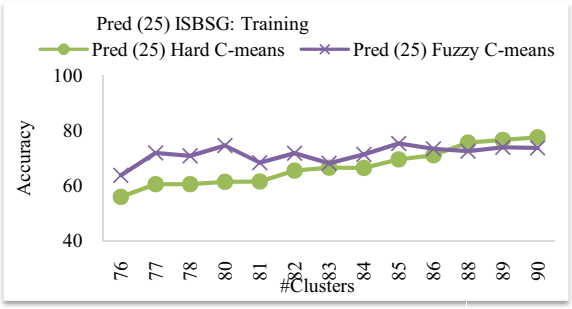

Fig. 7. Comparison of the Preds(25) of an RBFN using the hard or fuzzy C-means for the ISBSG dataset.

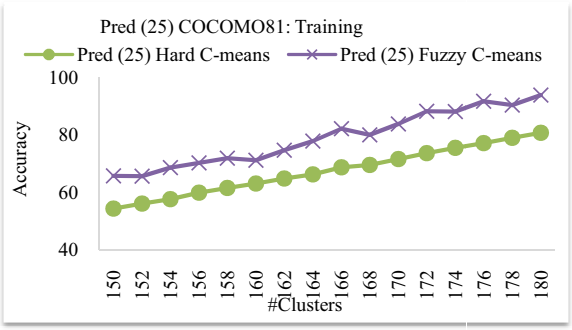

Fig. 8. Comparison of the Preds(25) of an RBFN using hard or fuzzy C-means for the COCOMO81 dataset.

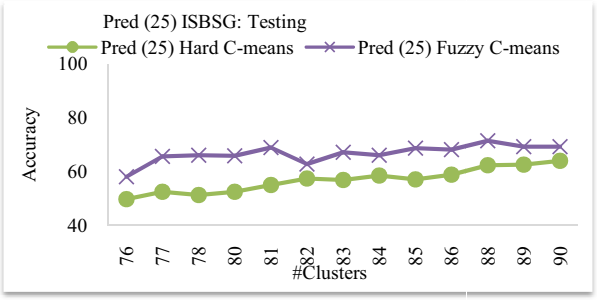

Fig. 9. Comparison of the Preds(25) of an RBFN using the hard or fuzzy C-means for the ISBSG dataset.

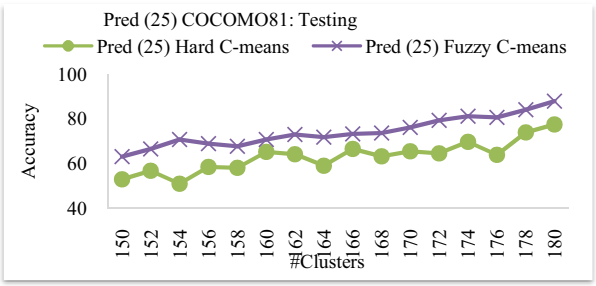

Fig. 10. Comparison of the Pred(25) of an RBFN using the hard or fuzzy C-means for the COCOMO81 dataset.

To conclude this section, Table 5 shows the lowest number of clusters having the acceptable accuracy of an RBFN using the hard C-means compared with that one of an RBFN when using the fuzzy C-means obtained from training and testing sets. Moreover, the minimum number of clusters for fuzzy C-means is almost the same either for training or testing sets whereas it is higher for training sets than that of testing sets for hard C-means. Indeed, the fuzzy concept allows the middle layer of an RBFN to perform a fuzzy classification of each project and hence the project may belong to several clusters. This is why even when the number of clusters is set to the maximum, an RBFN based on fuzzy C-means avoids the well-known over-fitting problem. By contrast, when the number of clusters is set to the maximum for an RBFN based on hard C-means, its middle layer generates a maximum number of clusters (e.g. each cluster contains one project) and hence each project only belongs to his cluster; so, there is a high risk of an over-fitting. This is why an RBFN based on fuzzy C-means generates acceptable values of Pred(25) on training and testing sets with almost the same minimum number of clusters allowing hence best performance in learning and generalization. However, an RBFN based on hard C-means generates acceptable Pred(25) but with a higher required number of clusters in testing than that one needed in training. This means a best performance in learning than in generalization.

TABLE V. LOWEST NUMBER OF CLUSTERS WITH ACCEPTABLE ACCURACY (PRED(25))

	Hard C-means		**Fuzzy C-means**	
Datasets	***Training Cluster #***	***Testing Cluster #***	***Training Cluster #***	***Testing Cluster #***
ISBSG	77	88	76	77
COCOMO81	150	160	150	150

VI. CONCLUSION

In this paper, we have empirically evaluated the RBFN designed with either hard or fuzzy C-means for software effort estimation. This study used a Cross-validation process on two historical datasets: ISBSG and COCOMO81. The Cross-validation generates for each dataset 10 collections; each one is composed of training and testing sets. The evaluations of an f RBFN based on hard/fuzzy C-means are carried out several times

on each training/testing sets of a given dataset/collection. The results obtained confirms the findings with the ISBSG and COCOMO81 datasets: (1) RBFN networks based either on hard or fuzzy C-means are a promising technique for software effort estimation; (2) RBFN using fuzzy C-means generates more accurate estimates than an RBFN using hard C-means in the testing phase. It is the same in learning phase especially when decreasing the number of clusters; (3) An RBFN using fuzzy C-means performs better with almost the same number of clusters whenever in learning or in testing phases. By contrast, an RBFN using hard C-means required more clusters in testing than in learning to generate accurate estimates. This due to the fact that fuzzy C-means tolerates imprecision than hard C-means and hence may allow the RBFN models to avoid the over-fitting problem.

REFERENCES

[1] M. Jorgensen, and M. Shepperd, "A Systematic Review of Software Development Cost Estimation Studies", IEEE Transactions on Software Engineering, Vol. 33, No. 1 (2007) 33-53.

[2] A. Idri, F. Z. Amazal, A. Abran, "Analogy-based Software Development Effort Estimation: A systematic and Review Study", Information and Software Technology, Vol. 58, Elsevier, 2015, pp. 206-230

[3] J. Wen, S. Li, Z. Lin, Y. Huc, C. Huang, Systematic literature review of machine learning based software development effort estimation models, Inf. Softw. Technol. 54 (1) (2012) 41–59.

[4] T. Halkjelsvik, M. Jørgensen, "From origami to software development: a review of studies on judgment-based predictions of performance time", Psychol. Bull.138 (2) (2012) 238-271.

[5] A. R. Gray, and S. MacDonell, and M. Shepperd, "Factors Systematically Associated with Errors in Subjective Estimates of Software Development Effort: The Stability of Expert Judgment", in 6th IEEE International Software Metrics Symposium (1999) 216-227.

[6] B.W. Boehm, "Software Engineering Economics," Prentice-Hall (1981).

[7] M. Jørgensen, A review of studies on expert estimation of softwaredevelopment effort, Journal of Systems and Software (2004) 37–60.

[8] G. R. Finnie, G. Witting, and J.M. Desharnais, "A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models, Systems and Software," Vol. 39, No. 3 (1997) 281-289.

[9] A. Idri, A. Abran, and S. Mbarki, "An Experiment on the Design of Radial Basis Function Neural Networks for Software Cost Estimation," in 2nd I EEE International Conference on Information and Communication Technologies: from Theory to Applications, Vol. 1 (2006) 230-235.

[10] K. Srinivasan, and D. Fisher, "Machine Learning Approaches to Estimating Software Development effort," IEEE Transactions on Software Engineering, Vol. 21, No. 2 (1995) 126-137.

[11] M. Shepperd and C. Schofield, "Estimating Software Project Effort Using Analogies," Transactions on Software Engineering, Vol. 23, No. 12 (1997) 736-747.

[12] A. Idri, and T. M. Khoshgoftaar, and A. Abran, "Investigating Soft Computing in Case-based Reasoning for Software Cost Estimation," Engineering Intelligent Systems for Electrical Engineering and Communications, Vol. 10, No. 3 (2002) 147-157.

[13] R. W. Selby and A.A. Porter, "Learning from examples: generation and evaluation of decision trees for software resource analysis," IEEE Transactions on Software Engineering, Vol. 14, No. 12 (1988) 1743-1757.

[14] M.O. Elish, "Improved estimation of software project effort usingmultiple additive regression trees," Expert Systems with Applications, Vol. 36, No. 7 (2009) 10774–10778.

[15] V. Sharma and H. K. Verma, "Optimized Fuzzy Logic Based Framework for Effort Estimation in Software Development," Computer Science Issues, Vol. 7, Issue 2, No. 2 (2010) 30-38.

[16] M. W. Nisar, Y. J. Wang, and M. Elahi, "Software Development Effort Estimation Using Fuzzy Logic – A Survey," in 5th International Conference on Fuzzy Systems and Knowledge Discovery (2008) 421-427.

[17] K.V. Kumar, V. Ravi, M. Carr, N.R. Kiran, "Software development cost estimation using wavelet neural networks," Journal of Systems and Software, vol.81 (2008) 1853–1867.

[18] H. Park, S. Baek, "An empirical validation of a neural network model for software effort estimation," Expert Systems with Applications, vol.35, No. 3 (2008) 929–937.

[19] R. Setiono, K. Dejaeger, W. Verbeke, D. Martens, B. Baesens, "Software effort prediction using regression rule extraction from neural networks", Proceedings of the 22nd International Conference on Tools with Artificial Intelligence, Arras, France, Vol. 2 (2010) 45–52.

[20] V. Khatibi. B, et al., "Neural networks for accurate estimation of software metrics," International Journal of Advancement in Computing Technology, Vol. 3 (2011) 54-66.

[21] J. Kaur, et al., "Neural network-a novel technique for software effort estimation," International Journal of Computer Theory and Engineering, Vol. 2 (2010) 17-19.

[22] I. K. Balich and C. L. Martin, "Applying a feedforward neural network for predicting software development effort of short-scale projects," in Software Engineering Research, Management and Applications (SERA) (2010) 269-275.

[23] I. Attarzadeh and S. H. Ow, "Software development cost and time forecasting using a high performance artificial neural network model," in Intelligent Computing and Information Science. Vol. 134, R. Chen, Ed., ed: Springer Berlin Heidelberg (2011) 18-26.

[24] A. Idri, A. Zahi, E. Mendes, and A. Zakrani, "Software Cost Estimation Models Using Radial Basis Function Neural Networks," in International Conference on Software Process and Product Measurement (2007) 21-31.

[25] A. Idri, A.Zakrani and A. Zahi, "Design of Radial Basis Function Neural Networks for Software Effort Estimation," in International Journal of Computer Science Issues (IJCSI),Vol. 7,Issue 4, No 3 (2010) 21- 31.

[26] A. Zakrani and A. Idri, "Applying Radial Basis Function Neural Networks Based on fuzzy Clustering to estimate web Applications Effort," in International Review on Computers and Software (I.RE.CO.S), Vol. 5, No 5 (2010).

[27] A. Idri, A. Hassani and A. Abrain, "Assessing RBFNBased Software Cost Estimation Models," In the 25th International Conference on Software Engineering and Knowledge Engineering (SEKE 2013) (Boston, MA, USA, June 27-29, 2013): Proceedings of the Twenty- Fifth International Conference on Software Engineering & Knowledge Engineering (2013) 483-487.

[28] ISBSG, International Software Benchmarking Standards Group, Data Release 8 Repository, (2003) http://www.isbsg.org.

[29] J. Moody and C. Darken, "Fast Learning in Networks of Locally- Tuned Processing Units," Neural Computing, Vol. 1 (1989) 281-294.

[30] A. Idri, L. Kjiri, A. Abran, 'COCOMO Cost Model Using Fuzzy Logic', 7ème International Conference On Fuzzy Theory and Technology, Atlantic City, NJ, 28 February-03 March, (2000). pp.219-223

[31] U.Gupta, M.Kumar, "Software effort estimation through clustering techniques of RBFN network," in IOSR Journal of Computer Engineering (IOSR-JCE),Vol. 14,Issue 3 (2013) 58-62.

[32] N. Benoudjit and M. Verleysen, "On the Kernel Widths in Radial-Basis Function Networks," Neural Processing Letters, Vol. 18, No. 2 (2003) 139-154.

[33] F. Schwenker and C. Dietrich, "Initialisation of Radial Basis Function Networks Using Classification Trees," Neural Networks World, Vol. 10 (2000) 476-482.

[34] S.Haykin, "Neural Networks: A comprehensive Foundation," Prentice Hall (1998).

[35] X. L. Xie and G. Beni, "A validity Measure for Fuzzy Clustering," IEEETrans. Pattern Anal. Mach. Intell, Vol. 13 (1991) 841-847.