IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 14, Issue 3 (Sep. - Oct. 2013), PP 58-62
www.iosrjournals.org

Software effort estimation through clustering techniques of RBFN network

Usha Gupta[1], Manoj Kumar[2]
[1]*(Assistant Professor, Department of Computer Science, MDU Rohtak, India)*
[2]*(Mtech. Scholar, Department of Computer Science, MDU Rohtak, India)*

Abstract: *Now a day's software cost/effort estimation is a very complex job to do. Several estimation techniques have been developed in this regard. This assessment of parameters like, time, cost, and number of staff required sequentially which in turn is to be done at an early stage. Constructive Cost model which is also known as COCOMO model was one of the best model to estimate the cost and time in person month of a software project. The estimation of cost and time supports the project planning and tracking, as well as also controls the expenses of software development process. The accurate effort estimation will lead to improve the project success rate. In this paper, the main focus is on finding the accuracy of estimation of effort/cost of software using radial basis function neural network (RBFN) incorporating ANN-COCOMO II which can be used for functional approximation. This model estimates the total effort of software development according to the characteristics of COCOMO-II along with radial basis clustering techniques. The RBFN network is much faster than other network because the learning process in this network has two stages and both stages can be made efficient by appropriate learning algorithms. The RBFN network uses COCOMO-II dataset for training.*
Keywords:- *COCOMO-2, RBFN, Clustering techniques, K-means, APC-III.*

I. Introduction

Effort estimation consist the assessment of how many hours of work needed and as well as how many workforces are required to develop a software project. Predicting the software development effort/cost with accurate results is a very challenging job for project managers. These estimation can be categorize in to three categories including Algorithmic Models (AM) [1] and Expert Judgment (EJ), and Machine Learning (ML) techniques. Among estimation techniques, COCOMO (Constructive Cost Model) is the most commonly used algorithmic cost modeling technique [2] because of its simplicity for estimating the effort in person-month for a project at different stages.

COCOMO uses the mathematical formulae to predict project cost estimation. The Constructive Cost Model (COCOMO) is the well known software effort estimation model based on regression techniques. The COCOMO model is used to calculate the amount of effort then on the basis of the effort calculated, the time, cost and number of staff required for software project are estimated. The Post-Architecture [3], [4] Level of COCOMO II consists of 17 cost drivers represented in the form of project attributes, programmer abilities, developments tools, and etc. These cost drivers and scale factors for COCOMO II are then rated on a scale from Very Low to Extra High in the same way as in COCOMO 81 dataset. The effort calculation formulas for COCOMO II post architecture level are calculated as given below [4]:

$$Effort = A * (size)^{SF} * \prod EM$$

Where
SF= Scale Factor usually lies between 1 and 1-5.
EM= Effort Multiplier
A: Multiplicative Constant depends upon types of software developed
Size: Size of the software measures in terms of KSLOC (kilo source line of code).
The scale factors (SF) are based on a significant source of exponential variation on a project's effort or productivity variation. COCOMO is the most commonly used model because of its simplicity for estimating the effort in person-month for a project at different stages.

An RBFN architecture [2], [5], [6] configured for software development effort 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. The activation functions of each middle layer neurons are usually the Gaussian function. 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.

II. constructive cost model (COCOMO)

COCOMO- Post Architecture model determines the efforts (in Person-Months) required for a project based on software project's size in KSLOC (Kilo Source Line Of Code) as well as other cost factors known as scale factors and effort multipliers. The estimated effort in person-month is given below [4]:

$$PM = A.(Size)^{1.01+\sum_{i=1}^{5} SF_i}.\prod_{i=1}^{17} EM_i$$

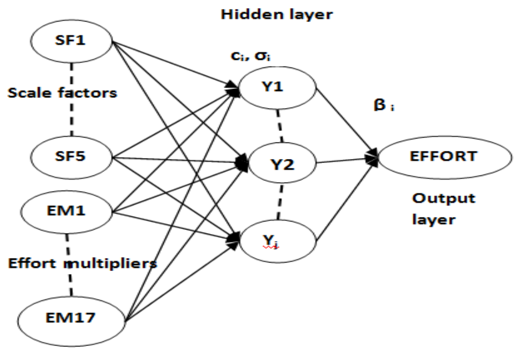
Where A is a multiplicative constant, and the set of SF (Scale Factor) and EM (Effort Multiplier) are defined in table given below [9]:

Symbol	Abbreviation	Name
SF1	PREC	Precedentedness
SF2	FLEX	Development Flexibity
SF3	RESL	Architecture and Risk Resolution
SF4	TEAM	Team Cohesion
SF5	PMAT	Process Maturity
EM1	RELY	Required Software
EM2	DATA	Database Size
EM3	CPLX	Product Complexity
EM4	RUSE	Required Reusability
EM5	DOCU	Documentation match to life-cycle needs
EM6	TIME	Time Constraints
EM7	STOR	Storage Constraints
EM8	PVOL	Platform Volatility
EM9	ACAP	Analyst Capability
EM10	PCAP	Programmer Capability
EM11	AEXP	Application experience
EM12	PEXP	Platform Experience
EM13	LTEX	Language and Tool Experience
EM14	PCON	Personnel Continuity
EM15	TOOL	Use of Software tools
EM16	SITE	Multi-site Development
EM17	SCED	Required Development Schedule

In this paper, the main focus is on assessing the accuracy of the estimation using RBFN network. The aim of this study is to check if RBFN network can be used for prediction of effort on the basis of effort multipliers, size of the project, scale factor used in project development.

III. Radial Basis Function Neural Network

An RBFN architecture [2] configured for software development effort 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, plus, eventually, one bias neuron. 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 [7]. This procedure initially distributes the respective fields of hidden layer neurons across the space of input variables.



IV. Clustering Algorithms for RBFN network

As in general, clustering is a method of finding the most similar groups of given data, which means that the data belonging to one cluster are the most similar, and data belonging to the different clusters are the most dissimilar ones. Clustering techniques have been used in many applications like: medicine, biology, pattern recognition etc. There are two types of clustering techniques briefly explained in this paper are as below:

- ***K-means***
- ***APC-III.***

K-means clustering algorithm has many successful applications such as pattern recognition and data compression. It is a multi-pass and time-consuming clustering algorithm.

4.1 K-means algorithm

This algorithm helps in determining the receptive fields at hidden layer of a Radial Basis Neural Network for estimating the cost of any software. In this algorithm, total vectors or datasets (say 'N') of COCOMO-2 model are partitioned in to clusters (say 'c').The objective of this algorithm is to find out the centers of the clusters known as centroids by minimizing the distance (or dissimilarity) between clusters. As per the calculations, this approach is very time consuming and also multi pass process.

This approach starts with defining the total number of clusters in advance before estimating the cost of software. This K-means approach [2], [6], [7] can be used with two variants/ criteria's in order to measure the coherence of clusters as:

- ***J function value***
- ***Dunn's validity index function value***

4.1.1 K-means with J function value criteria (J value):

In J function value criteria, the data points or datasets are assigned to different clusters on the basis of Euclidean distance values. The algorithmic approach is explained below:

Algorithm: RBFN_K-means J-value (c, x, C, J)

(Here c=total number of clusters
x=dataset from COCOMO-2 dataset model.
C=cluster.
J=dissimilarity or distance function).

Step 1: start the K-means procedure for radial basis function for neural network with J function value.
Step 2: define the total number of clusters desired, say 'c'.
Step 3: initialize the centers of clusters to 1, say $c_{i,,}$ where 'c' is the total number of clusters. And 'i' is used for iteration and for i=1 to c.
Step 4: now compute the Euclidean distance between x_j and c_i, for j=1 to N, and i=1 to c. here x_j represents a dataset from dataset collection of COCOMO-2 model, j indicates iteration for each dataset.
Step 5: now according to the Euclidean distance values, assign the x_j to the most closer cluster C_i on the basis of less distance value (compare the distance values for x_j to each and every cluster c_i).
Step 6: now again recalculate the centers of each and every cluster say 'c_i'.
Step 7: now calculate the distance or dissimilarity function say 'J' as given below:

$$J = \sum_{i=1}^{c} \sum_{xj \in Ci} d(xj, ci)$$

Here d(xj,ci) is the measure of distance value between i^th center (ci) and the k^th data point of dataset of COCOMO-2 model.

All for the sake of simplicity this d value can be taken as Euclidean distance value and hence the J function value can be represented as given below:

$$J = \sum_{i=1}^{c} \sum_{xj \in Ci} \|xj - ci\|^2$$

Step 8: check if the improvement is below a certain threshold value, and then stop the iteration.
Step 9: else go to step 3.
Step 10: exit.

The main goal of this approach is to minimize the J function value.

4.1.2 Dunn's validity index function value (D value):

In case of Dunn's validity index function, the data points are assigned to the clusters on the basis of inter-cluster distance as well as intra-cluster distance. The algorithmic approach [5], [7] is explained below:

Algorithm: RBFN_K-means D-value (c, x, C, D)
(Here c=total number of clusters
x=dataset from COCOMO-2 dataset model.
C=cluster.
D=Dunn's validity index function value).
Step 1: start the K-means procedure for radial basis function for neural network with Dunn's validity index.
Step 2: define the total number of clusters desired, say 'c'.
Step 3: now compute the Euclidean distance between two randomly selected clusters say C_i and C_j, for i=1 to c, and j=2 to c. The distance $d(C_i, C_j)$ is referred to as the minimum inter-cluster distance as given below:
$d(C_i, C_j)=min_{xi\in Ci,xj\in cj}\, d(x_i, x_j)$
Step 4: now compute the Euclidean distance with in a randomly selected cluster say C_k for k=1 to c. The distance $d(C_k)$ is referred to as the maximum intra-cluster distance calculated as below:
$d(C_k)=max_{xi,xj\in Ck}\, d(x_j, x_k)$
Step 5: now calculate the dunn's validity index say 'D' on the basis of inter-cluster distance and intra-cluster distance values. This D value can be calculated by the formula given below:

$$D = min_{1\leq i\leq c}\left[min_{i+1\leq j\leq c-1}\left[\frac{d(Ci,Cj)}{max_{1\leq k\leq c}(d(Ck))}\right]\right] \quad \ldots(1)$$

Step 7: this D value calculated from equation 1 is used as an idea of identifying whether the clusters are compact and well separated or not.
Step 8: exit.

The main goal of the measure of the D value is to maximize the inter-cluster distances and minimizing the intra-cluster distances.

As per the algorithm or procedure given above, it is very much obvious that the performance of the RBFN K-means algorithm depends only upon the initial positions of the cluster centers as the number of clusters is taken as an initial value to start off the procedure. And the data points or the datasets are assigned to these clusters one by one. So there lies no guarantee for the optimality of the solutions for estimating the cost of software. And hence it is said to be a quite lengthy approach, also known as multi pass approach.

4.2 APC-III approach:

This is a one pass clustering algorithm. This approach works by calculating the constant radius of cluster through the following formula:

$$R_0 = \alpha * \frac{1}{N} * \sum_{i=1}^{N} min_{i\neq j}\left(\|P_i - P_j\|\right)$$

Where
N=number of historical software projects
α=predetermined constant
R_0=radius of each cluster which in turn controls the number of clusters provided.

APC-III approach [5], [7] works by generating clusters for each dataset. As the classification generated by the APC-III algorithm depends on the number of α that defines the radius R_0. The number of clusters generated is inversely proportional to the radius.

$$Number\ of\ clusters \propto \frac{1}{R_0}$$

As the value of R_0 decreases, the numbers of clusters generated are also increases. In short, APC-III approach generates many clusters if R_0 is small and few numbers of clusters if it is large.

```
Algorithm: RBFN_APC-III (c, C, N, $P_i$, $R_0$)
(Here c=total number of clusters
C=center of cluster.
N=number of historical software projects
$P_i$ =dataset from COCOMO-2 dataset model.
$R_0$=radius of clusters).
Step 1: start the APC-III procedure for radial basis function for neural network.
Step 2: set the total number of clusters initially to 1, i.e. c=1.
Step 3: now set $P_1$ as the center of this cluster $C_1$, here $P_1$ represents the first software project in the datasets.
Step 4: now for i=2 to N (N=total number of historical software projects), repeat the following steps.
Step 4(a): for j=1 to c (c=number of clusters)
    i.   Calculate the $d_{ij}$ ($d_{ij}$ is the Euclidean distance of $P_i$ and $c_j$, $c_j$ is the center of the cluster $C_j$).
    ii.  now check if $d_{ij} < R_0$ then,
            i.   Add the $P_i$ dataset into $C_j$ and correspondingly adjust the center of $C_{j.}$
            ii.  Exit from the loop.
Step 4(b): else, if $P_i$ is not included in to any of the clusters then,
    i.   Create a new cluster that contains $P_i$ as a center.
Step 5: stop the criteria.
```

According to Hwang and Bang, the APC-III is quite an efficient technique for constructing the middle layer of an RBFN because it can finish making clusters by going through all the training patterns only once. However it still proves to be dependent on the order of the data presentation.

V. Conclusion

In this paper, there are three most popular clustering approaches were presented to predict the software cost estimation. In one hand COCOMO which has been already proven and successfully applied in the software cost estimation field and in other hand the Radial Basis Function Neural Network that has been extensively used in incorporating COCOMO estimation and have proven its strength in estimating the cost. Furthermore, the APC-III clustering algorithm is used to estimate the efforts on the basis of historical projects and by calculating the radius function. On the other hand, the K-means clustering algorithm is used to predict the mean value which is useful in Gaussian function for network. To get accurate results the proposed network depends only on adjustments of weights from hidden layer of network to output layer of RBFN network. RBFN network with the APC-III clustering algorithm performs better, in terms cost estimation accuracy as compared to that of K-means approach because of its complexity and simplicity. This work can be extended by integrating with Fuzzy C-means clustering algorithm as well as with genetic algorithm approach.

References:

[1] A New Approach For Estimating Software Effort. Using RBFN Network. Ch. Satyananda Reddy, P. Sankara Rao, KVSVN Raju, V. Valli Kumari, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.7, July 2008.

[2] A new approach of Software Effort Estimation Using Radial Basis. Sriman Srichandan Balasore College of Engineering and Technology,Balasore,Odisha ,India, International Journal on Advanced Computer Theory and Engineering (IJACTE), ISSN (Print) : 2319 – 2526, Volume-1, Issue-1, 2012, pp. 113-120.

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

[4] K. Molokken, and M. Jorgensen, "A Review of Surveys on Software Effort Estimation", in International Symposium on Empirical Software Engineering, 2003, pp. 223-231.

[5] G. R. Finnie, and 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, pp. 281-289.

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

[7] A. Idri, and A. Zahi, and 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, pp. 21-31.