***SFNE: Swarm Intelligence based Functional Link Fuzzy Neural Estimator for Software Development Effort Estimation***

Tirimula Rao Benala1, Rajib Mall2

1Department of Information Technology

Jawaharlal Nehru Technological University Kakinada, University College Of Engineering Vizianagaram-535003, India

b.tirimula@gmail.com

2 Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

rajib@cse.iitkgp.ernet.in

Rohitha Bandarupalli

Department of Information Technology

JNTUK, University College of Engineering Vizianagaram

Vizianagaram-535003, India

rohitha.bandarupalli@gmail.com

Abstract— Computational intelligence – a hybrid system of machine learning, swarm intelligence, evolutionary computation, and fuzzy logic- is considered an emerging topic for frontier research in the field of software development effort estimation (SDEE). It is expected to improve the accuracy of software effort estimation. We investigate improvements in estimation of development effort by using Computational Intelligence models with (1) Particle Swarm Optimization (PSO)- Functional Link Artificial Neural Network (FLANN) (2)Interval Type 2 Fuzzy logic system(IT2FLS)-FLANN-PSO (3) Active Learning algorithm (ACTIVE)-FLANN-PSO (4) IT2FLS-ACTIVE-FLANN-PSO and analyzing efficacy of different Semi-Automatic Intelligent Estimators. We name this as Swarm Intelligence based Functional Link Fuzzy Neural Estimator for software development effort estimation (SFNE). Our SFNE method includes (1) QUICK - used for dataset reduction (2) The reduced dataset is computationally processed using FLANN (3) To reduce the uncertainty in crisp sets we have opted IT2FLS(4) The tuned software effort output error is optimized using Particle Swarm Optimization. An extensive simulation study is performed on 6 datasets from PROMISE repository that show the effectiveness of our proposed SFNE techniques. The experimental results demonstrate that SFNE exhibits promising results.

Keywords—Software cost estimation,Functional Link Artificial Neural Network,Fuzzy logic system,Interval Type-2FLS,Particle Swarm Optimization

# Introduction

Software Cost Estimation has been identified as one of the three great challenges in computer science. (Fred Brooks 2003). Every year either a significant number of new applications are produced or existing applications are modified. According to Standish group Chaos Report 2013, 18% of the software projects fail due to poor software cost estimation practices. In this context, our research objective is to devise a suitable SCE technique. Estimated software cost serves as the basis for almost every project planning activity. Therefore inaccuracy in cost estimation can lead to repercussions. Two prime issues in Software cost estimation is over and under estimation. Underestimating a project leads to allocation of lesser staff, design of short schedules, and production of low-quality deliverables. In contrast, overestimating a project leads to either customer cancellation of the project, or leads allocation of more resources to the project than needed, causing under productivity.

A major challenge in cost estimation is accurately estimating the size and cost of the software artifact to be developed at the planning phase, that is long before the project development work actually begins. Due to the inherent uncertainties present in any project, most current techniques used for software cost estimation tend to be inaccurate at the early stages of software development and only become better as the project progresses (Boehm and Valerdi, 2006). The most of the traditional parametric software effort estimation models are based on multiple regression approach. The inherent property of these models is to accurately predict the effort via calibration of actual data collected from completed software projects. Examples of popular parametric effort prediction models include Constructive Cost Model (COCOMO) (Boehm, 1981), Constructive Cost Model II (COCOMO II) (Boehm, 1995), and Software Life Cycle Management (SLIM) (Putnam, 1978). These models can have serious difficulties when used on software engineering data that is usually scarce, incomplete, and imprecisely collected. (Boehm, 2000;S Devnani-Chulani, 1999). The problems with traditional parametric models have emphasized the need for composite techniques. To alleviate these problems, an effort estimator based on Computational intelligence (CI) techniques – synergy of neural networks, evolutionary computation (EC), Fuzzy system (FS) and Swarm Intelligence (SI) - for building software models is discussed in this paper. The popularity of CI techniques lies in the fact that they do not require precise models for evaluating the cost function. (Witold Pedrycz 2002, Benala T.R., et al., 2012). We have used the hybrid computational intelligence model comprising of swarm intelligence based functional link neural system integrated with interval type-2 fuzzy logic system (known as SFNE) to build software development effort estimation model. Interval Type-2 FLS is a simplified version of Type-2 FLS. A type-2 fuzzy set (Uncu, O., & Turksen, I. B., 2007) is more capable to incorporate uncertainties compared to type-1 fuzzy logic system. Due to computational complexity of type-2 fuzzy logic system, Liang and Mendel proposed interval type-2 FLS, i.e. a simplified version of Type-2FLS which possesses all the advantages of Type-2FLS sans its computational complexity. This has all been dealt in section 2. However, a detailed discussion can be seen in ref. (Mendel, J. M., John, R. I., & Liu, F., 2006). Further, interval Type-2FLS used in this paper belongs to TSK (Takagi-Sugano-Kang) type (Begian, M. B., Melek, W. W., & Mendel, J. M., 2008, Juang, C. F., & Tsao, Y. W., 2008) yielding an easier defuzzification procedure for predicting software effort.

In section II, the framework of ABE is studied. The LS-SVM is discussed in section III, and finally, the architecture of the proposed method is represented in section IV. Section V and Section VI depicts the evaluation criteria and experimental analysis respectively. The conclusion is in section VII.

# RELATED WORK

Several research results have been reported in application of computational intelligence in software cost estimation. In this section, we briefly review these research efforts.

Muzaffar, Z., & Ahmed, M. A. (2010) have presented a study on fuzzy logic system factors that impact the accuracy of software development effort prediction.. Ahmed, M. A., & Muzaffar, Z. (2009) have investigated an effort prediction framework that is based on type-2 fuzzy logic to allow handling imprecision and uncertainty inherent in the information available for effort prediction. Xu, Z., & Khoshgoftaar, T. M. (2004) have used an innovative fuzzy identification cost estimation modeling technique to deal with linguistic data, and automatically generate fuzzy membership functions and rules. In Azzeh, M., Neagu, D., & Cowling, P. (2009) two important research areas in EBA are addressed: software projects similarity measurement and attribute weighting by combining the advantages of Fuzzy Set Theory and Grey Relational Analysis. Sheta, A. (2006) has used Takagi-Sugeno (TS) technique to develop fuzzy models for two nonlinear processes. Lee, J. et. al. (2011) has proposed fuzzy size estimation procedure for goal-driven use case model based on UCP using fuzzy theory. In 2013 Ekrem et al. proposed Active learning and Effort Estimation.

The use of Functional link artificial neural network (FLANN) for software development effort estimation was reported by Benala T.R. et al. (2009). In their work the authors have done empirical validation of the FLANN based software development effort estimation model using COCOMO’81 dataset of PROMISE repository test suit. Later on, the authors have proposed number of approaches for software development effort estimation using FLANN. In 2012 they proposed three approaches (Benala T.R. et al., 2012a, 2012b, 2012c). The proposed methods are validated on ﬁve real time datasets from PROMISE repository test suit. Their approach reported promising results over conventional FLANN, support vector machine regression (SVR), radial basis function (RBF), classiﬁcation, and regression trees (CART).In second approach; they proposed a genetic algorithm for optimizing functional link artiﬁcial neural networks for software development effort estimation. In their third approach they used fuzzy clustering and functional link artiﬁcial neural networks for software development effort estimation.

Benala T.R. et al. (2013) used particle swarm optimization (PSO) to optimize the feature weights of FLANN. The framework is annotated as PSO-FLANN. PSO-FLANN showed promising results In 2014 Benala T.R. et al., proposed software effort estimation using data mining techniques. In 2015 Benala T.R. et al. proposed Software Effort Estimation Using Functional Link Neural Networks Tuned with Active Learning and Optimized with Particle Swarm Optimization. This work advocated a new learning model based on the collaborative effort of active learning and particle swarm optimization (PSO) in functional link artificial neural networks (FLANNs) to estimate software effort.

We could not find any work to develop a hybrid system using these computational intelligence techniques. In this paper, we have incorporated interval type-2 FLS, Active learning, and PSO to evolve HONs. Furthermore we have empirically validated the proposed model with six benchmarking datasets from PROMISE repository test suit. The rest of the paper has been organized as follows. In section II we have discussed background material. Section III provides our proposed software development effort estimation model SFNE. In section IV we have presented the experimental framework, comparative performance with other prediction models like ANN and FLANN with gradient descent, RBF. Section V concludes the article.

# preliminaries

In this section, we first discuss the different types of fuzzy systems. Subsequently, we discuss the architecture of FLANN.

## Type 1 Fuzzy logic system vs. Interval type 2 fuzzy logic system.

Consider the transition from crisp set to fuzzy set. When it is difficult to determine the membership of an element in a set as 0 or 1, fuzzy set of type-1 is used. Similarly when the circumstances are so fuzzy that we cannot determine the membership grade even as a crisp number in [0, 1], fuzzy set of type-2 are used. (Méndez, G. M., et al., 2014; J.M.Mendel, 2001). Both type-1 and type-2 fuzzy logic systems are considered as state-of-art tools to handle uncertainty in complex real world problems. The differentiator in the structure of T2FLS as compared to T1FLS is large number of design degree of freedom. The structure of a general T2FLS is shown in Fig. 2. The output processor of a T1FLS maps a T1 FS to a crisp number whilst a T2FLS has two components in the output processor, the ﬁrst component is a type reduction that transforms a T2 FS into a T1 FS and the second one is the defuzziﬁer that transforms a T1 FS into a crisp number. A general T2FLS is computationally expensive and has complicated implementation compared to a T1FLS. A special case of T2FLS, interval type-2 FLS (IT2FLS) has been widely used for reduced computational burden (Mendel, John, & Liu, 2006; Mendel & Liu, 2013; Nguyen, T., et al., 2015). Type-1 fuzzy sets represent belongingness of a crisp value of a base variable in a fuzzy set A characterized by a crisp membership function that takes on values in the interval [0, 1]. Such a set may be represented as:

(1)

Thus they cannot exhibit the uncertainties due to imprecision in identifying membership functions. To overcome the limitation, type-2 FLS (Castillo, O. et al., 2007). has been introduced to minimize the effect of the uncertainty in the rule base and used in the areas like modeling and control, data mining, system identification, forecasting, computer vision, pattern recognition etc ( Chakravarty, S. et al. 2012).

The representation of a general Type-2 and IT2FLS are different from type-1 FLS by a tilde symbol. For example, If denotes a type-1 fuzzy set, then denotes interval type-2 fuzzy set or type-2 fuzzy set. A type-2 fuzzy set, denoted by, is characterized by membership function (MF) , where and that is

(2)

Where

The amplitude of a secondary membership function is called secondary grade. In Eq. (1), for is a secondary grade.When the values of secondary grade are the same and equal to 1, then it results in an interval type-2 membership function. Thus if, then is an interval type-2 fuzzy set.

IT2FLS is bounded by two membership functions: lower membership function and upper membership function. Each of the two membership functions can be represented by a type-1 fuzzy set membership function. The area between these two membership values represents the foot print of uncertainty (FOU), which is the union of all primary membership functions and consists of a bounded region shown in fig. 1.

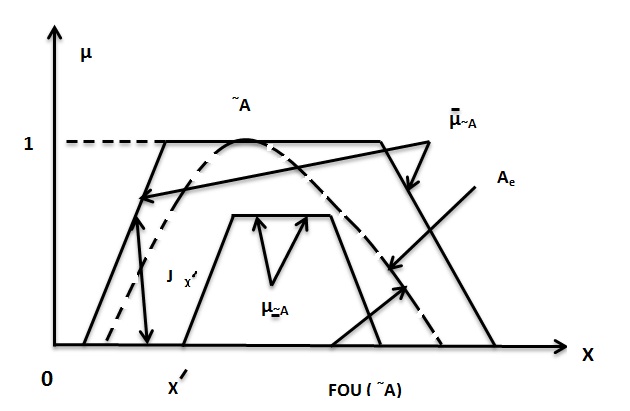


Fig. 1 IT2 FL and its associated quantities.

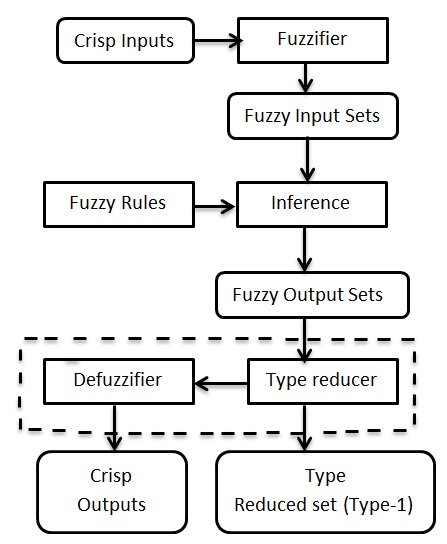


Fig. 2. A general Structure of type-2 FLS

# *Architecture of Functional Link Artificial Neural Network*

Functional Link Artificial Neural Network (FLANN) originally proposed by Pao is a novel single layer neural network with faster convergence rate and computationally efficient neural network model as compared to multilayer perceptron (MLP). The typical structure of FLANN is shown in figure 3. The nonlinearity in FLANN is introduced by orthogonal functional expansions (a.k.a basis functions). The commonly used basis functions are Chebyshev polynomial, Legendre polynomial, and power polynomial. The software effort estimation is functional approximation optimization problem. The goal of FLANN can be defined as selecting a basis function to learn the effort estimation function by approximating the function. The interpolation of the function is achieved by FLANN is the set of weights to be optimized to obtain the best approximate of.

The Chebyshev polynomial functional expansion is preferred as basis function in our work due to low error estimate characteristic. The input space is subjected to nonlinear transformation by basis function and feature space is created at high dimension. The feature space is multiplied by weight vector selected in the range [-0.5, +0.5]. The summation is fed to sigmoid function to predict the cost function. The optimal cost function is obtained by iteratively updating the weight vector. The weight vector is evolved by Particle Swarm Optimization(PSO) learning algorithm. (Benala T.R. et al., 2012a, 2012b, 2012c).

1. *Active Learning for Dataset Reduction:*

Active learning is introduced by Simon in 1974 (Dasgupta, S., 2011). The key idea behind active learning is to improve the performance of FLANN by choosing useful samples from Software effort estimation data set. QUICK, an active learning algorithm proposed by Ekrem Kocaguneli et al., has been utilized in this article. It identifies the essential content of dataset to be fed to the SFNE model for improving the estimation accuracy.The QUICK method has two important components --- synonym pruning and outlier pruning. At first the dataset is represented by a 2-d matrix. The rows represent project instances and columns represent the features or attributes. In synonym pruning the dataset is transposed and the similarity measure between attributes is calculated by Euclidean distance. After getting distance matrix, for similar values in each row ranks are assigned by incrementing by 1. Those attributes having similar neighbors represented by popularity index are removed. Next, the process of outlier pruning is initiated. The obtained matrix is transposed to get back to its original form. Now the rows will represent project instance and columns will represent features. The matrix contains only selected features from previous phase. Generate the distance matrix using Euclidean distance measure. Sort the rows based on the distance. The k-closet neighbors of another instance are defined to be popular. The popular ones are retained and the unpopular project instances are removed. Thus we obtain the most useful data samples to be fed to the next stage.

1. *Swarm Optimization Learning Algorithm.*

Particle swarm optimization is a population based multi agent stochastic algorithm proposed by Eberhart and Kennedy in 1995. In PSO, a potential solution is known as particle and a set of these solutions is a “Swarm”. Each particle’s state information is represented by its current position and current velocity in the search space. Set of randomly generated particles (known as trial solutions) flies through the D-dimensional search space towards the optimal solution over a number of iterations by utilizing its best position and global best particle state in each iteration.

In the D-dimensional design space, let the position and velocity vectors of particle for the dimension be respectively. At any sampling instance t, the velocity and position can be updated as

(3)

(4)

The symbol denotes point by point vector multiplication, the inertia weight; and, chosen uniformly and randomly in some interval [0, 2], are acceleration coefficients influence the maximum size of the step which a particle can take in one iteration. The state space exploitation is introduced by distributed vectors random numbers and in the range[0,1]. To insure optimal convergence the adaptive inertia weight strategy is adapted in our work. The inertia weight decrease as the generation increases to balance the exploration and exploitation trade off in the search space (Benala T.R. et al., 2015).



# *SFNE for Software Development effort prediction*

In this section we will discuss the proposed SFNE for Software development effort prediction. This section is divided into two subsections, namely a high-level for the **S**warm Intelligence based Functional Link **F**uzzy **N**eural **E**stimator for software development effort estimation (SFNE), illustration of experimental procedure for SFNE using sample example.

1. *High- level Algorithm of SFNE:*

SFNE is an integrated model which combines two computational models, namely FLANN and interval type-2 fuzzy logic system. Thus, it has the advantage of both the models. Further it incorporates active learning technique, namely QUICK as preprocessing step which helps to improve the prediction accuracy by feeding essential inputs to the system. The weight updation in FLANN is done by adaptive PSO. Hence, it always converges to global optima unlike FLANN with back propagation which may trap in local minima.

SFNE is a typically a five layer network as shown in figure 3. The layer one takes the input from the dataset and transmits to the next layer. The layer two initiates the interval type-2 fuzzy logic system (IT2FLS). Our method incorporates a variant of Wang and Mendel (1991) approach for generating fuzzy rules from normal data. The type-2 Gaussian membership function with uncertain mean is considered for the antecedent and consequent variable (s) and type-2 Gaussian with uncertain standard deviation is considered as the membership functions for the input in this work. An IT2FLS deals with two membership functions, namely lower and upper membership functions. Nodes in layer three receive degrees of associated rules from nodes of layer two. In the proposed model two rules are generated each having two values (lower and upper). Thus total output of this layer is four. Nodes in layer four are called consequent nodes. The outputs obtained from layer three and two local outputs of FLANN are considered as the inputs to this layer. The final layer is the output processing layer. It consists of two back to back components. The first component is type reducer. We have adapted Karnik and Mendel algorithm (Mendel, 2001) for type reduction. The final crisp output is obtained from the defuzzification component. The methodology is represented in algorithm 2.

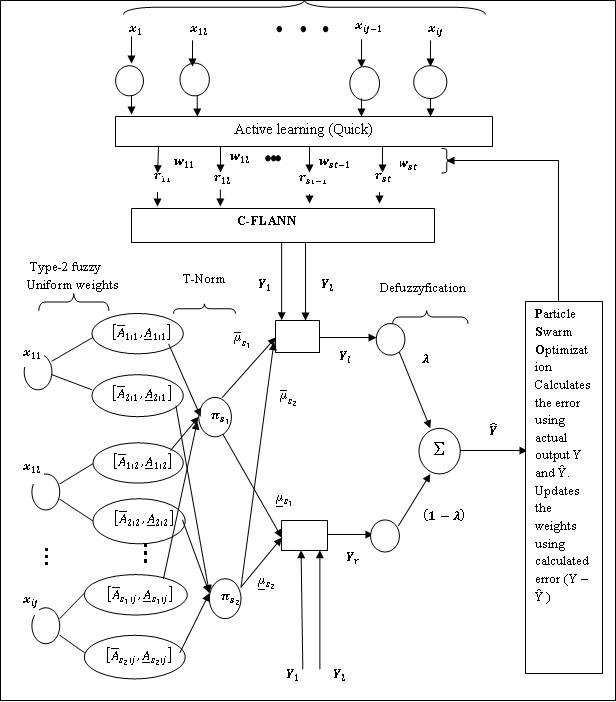
**

Fig. 3. Layered structure of SFNE using Mamdani Fuzzy Reasoning System

***Algorithm* :** Let us assume, D is the dataset, ND is the normalized Dataset. TrainDataset and TestDataset are the training and testing parts respectively, P is the data point of reduced size, L and H are lower and higher dimensions respectively, and TC is the termination criteria, O is the output layer, W is the weighted sum, E is the error, current best fitness value is CF. Fitness value is F, GB represents global best, PV is the particle velocity.

*Step 1: For each*

*Step 1.1:*

*Step 2: Divide ND 2/3rd parts into TrainDataset and 1/3rd part into TestDataset.*

*Step 3:*

*Returns upper and lower membership values of segment1 (s1) and segnent2 (s2)*

*Step 3: For each TrainDataset*

*Step 3.1: Apply*

*Step 4: For each P*

*Step 4.1: Map from L to H.*

*Step 5: For each particle initialize with small values from [-1, 1].*

*Step 6: While (!TC)*

*{*

*Apply*

*{*

*For each swarm*

*{*

*For each particle in the swarm*

*{*

*For each sample in the training sample*

*{*

*Calculate W, and send it as input to O.*

*The tuned values of Interval Type 2 fuzzy*

*are converted to type 1 fuzzy Using outputs (Y1 & Y2) in output layer*

*Then again defuzzyfy the type 1 values to crisp set using*

*Output=*

*Calculate.*

*}*

*Assign E to F.*

*If (F is better than CF)*

*{*

*Assign F to CF*

*}*

*Assign CF to GB*

*}*

*For each particle*

*{*

*Call Reduced () and Find PV.*

*Update Particle Positions*

*}*

*}*

*}*

*}*

1. *Experiment Design:*

In this study ,we have selected six PROMISE repository datasets for empirical tests. The performance evaluation metrics is presented in section A. The dataset is described in the section B. Additionally,al the cost estimation methods included in our experiments procedure are presented in section C. The results are analysed in section D.

1. *Performance Evaluation matrix:*

In this section, we present the evaluation metrics that we evaluate in the study. Performance metrics are important inorder to assess the efficacy of SDEE models. Several quality metrics have been used to evaluate the performance of estimation methods (Huang & Chiu, 2006). For allowing a more robust performance evaluation we have chosen following five performance measures: mean magnitude of relative error (MMRE), median magnitude of relative error (MdMRE) and PRED (0.25),Standardized Accuracy (SA) (Shepperd and MacDonnell, 2012)and Delta (Cohen, 1992).These evaluation criteria have been chosen as they are widely accepted benchmark metrics for performance evaluation in the software cost estimation literature.

When the PRED metric is defined as PRED (0.25)

(9)

Another measure, MAE which doesnot present asymmetry problems have been chosen as error indicator. The MAE is simply calculated by taking the average of absolute error(AE) as shown in equations 4 and 9. However, it is difficult to interpret as the residuals are not standardized. Later, Sheppred and Mc Donell came up with a new measure called as Standardized Accuracy(SA) as shown in

Eq 10 and to judge the effect size, following measure is suggested as shown in Eq 11. The essay measure is used mainly to test whether the prediction model in hand really outperforms a baseline of random guessing and generates meaningful predictions. If not so, we cannot even claim that this prediction model is meaningful. SA can be intyerpreted as the ratio of how much better a model is than random guessing, giving a very good idea of how well yi for the target case t by randomly sampling

The Lagrange multipliers areai ( i=1, 2, 3, ……, n)[2].

Here, the acquisition of Karush-Kuhn-Tucker conditions to solve the optimal parameters a = [a1, a2,....., an]T can be reported as

(7)

The elimination of parameters w and ḉi yields

image005 (8)

with I is identity matrix, image027= [1,1,.....1]T , a=[a1, a2,......, an]T , and the Mercer’s condition

image029 (9)

have been applied.

The following are the rewritten equations after solving the Lagrange coefficient a and model offset b from (7).

image007_(1)

To be specific,

image033 (10)

Substituting (9) into (7),

image035 (11)

Lastly, the LS-SVM classifier can be described as

image037 (12)

(5)

Hence, (6)

MdMRE ,also known as global measure of error, is defined as median of all BRE’s. It is less sensitive to outliers.

(7)

PRED(x) is defined as the percentage of predictions falling within the actual known value x, referred to as PRED is shown in Eq. ()

(8)

All the kernel functions ψ(x, y) should comply with the Mercer’s condition. In this study, the Radial Basis Function is considered as kernel function and the following equation shows the formula for RBF kernel function[11].

image041 (13)

The parameters γ and σ are the unspecified parameters.

# Analogy-Based System With Adjustment Mechanism

From the illustrations in section 2, this study synthesizes the ‘update,' which is generated by an adjustment mechanism, and the retrieved value, which is an unadjusted solution predicted by ABE, to form the target solution. Now, the extension of linear adjustment model, which put forth by Chiu and Huang in 2007 [14], can be noted as:

Ĉx = Ca + S(up, Ux) (14)

where S( . ) is a random function similar to the update, which is essential to transform the ABE prediction into target solution ( in our study, S( . ) is LS-SVM model ), up is the attribute vector of project x, Uk represents attribute matrix of K nearest analogies, and Ca denotes the unadjusted effort value from ABE (or retrieved solution)[1].

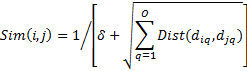
Our methodology incorporates two phases. Initially, it procures the ABE prediction and training would be done by LS-SVM to produce adjustment model. In the second stage, this trained model facilitates to induce the update, which adds on to ABE prediction to estimate the final estimation

## Phase I - Training

Fig 2 depicts the procedure of Phase I. T The adjustment model (LS-SVM) is trained by practicing leave-one-out cross-validation, which is similar to Jackknife approach. The following steps are accomplished for every project of the training set**.**

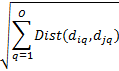
Step 1: In this step, *i*th project, which is treated as the latest project ready for estimation, is retrieved from training set while remaining projects are considered as past projects in the Analogy-Based Estimation (ABE) approach.

Step 2: Using similarity measure, the detection of K nearest similar projects from the past projects is done by the ABE approach. For the calculation of similarity function *Sim(i, j),* Euclidean distance is employed:

 (15)

δ=0.0001

image001_(3)

Where *I* denote the latest project, j indicated as individual past project, did represents *i*th attribute value of the *i*th project, djqis the *q*th attribute value of project j, O represents the overall figure of attributes in every project , and the constant δ=0.0001 presented as averting the condition when  = 0. In our similarity function, the un-weighted Euclidean distance is used to remove the effects of various attribute weights.

Now, the retrieved solution, which is un-weighted mean, for the *i*th project is generated after extracting the K analogies.

Step 3: The training process starts after generating the ABE prediction. The regularization parameter γ and kernel based parameter σ will have to be trained to produce the LS-SVM adjustment model. For that, the residuals between attributes of project *i* and attributes of its K similar projects are served as inputs to LS-SVM model, and the training target will be residual between the *i*th project actual effort values and the ABE prediction from its K analogies.

## Phase II – Predicting

In the predicting phase, Fig 3., a project x uses the trained LS-SVM model. For this, a group of K analogies be drawn from the trained data set by (14) to compute the similarities. As per above mentioned procedure, by using these analogies, the ABE system predicts the unadjusted solution. Now, the residuals of project i features and its K analogy features will be inputs to trained LS-SVM technique for getting the update. Lastly, the summation of ABE estimation and solution from adjustment model can be done to generate the predicted cost of the project i.

# Performance Evaluation Metrics

The evaluation metrics is essential to assess the performance of the regression models. It facilitates to know how well the generated prediction is close to the actual effort of a project. In this literature, Mean Balanced Relative Error(MBRE), PRED, and Mean Magnitude of Relative Error (MMRE) is implemented to analyze the performance of the current method.

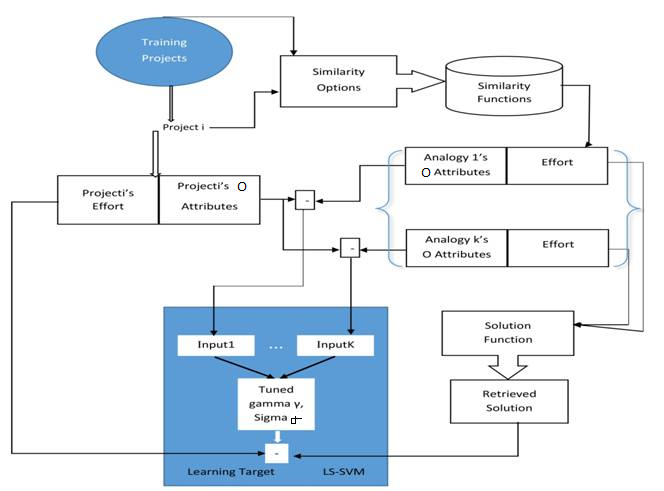


Fig 2 Training mechanism of the proposed method ABE-LS-SVM

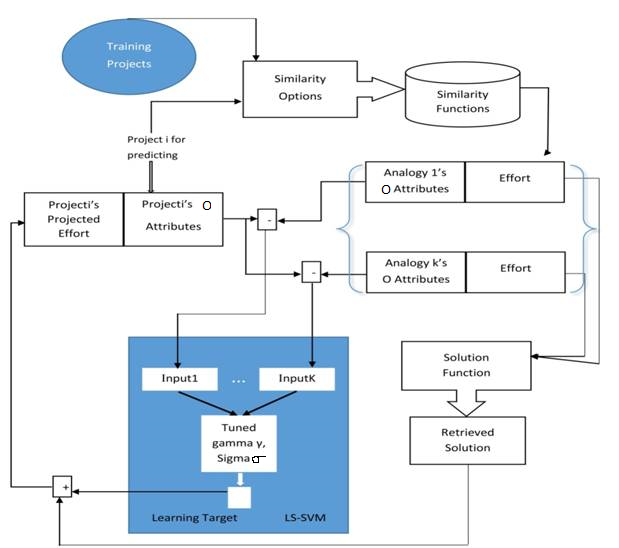


Fig 3 Predicting mechanism of the proposed method ABE-LS-SVM

* For papers in which all authors are employed by the US government, the copyright notice is: U.S. Government work not protected by U.S. copyright
* For papers in which all authors are employed by a Crown government (UK, Canada, and Australia), the copyright notice is: 978-1-5090-2807-8/16/$31.00 ©2016 Crown
* For papers in which all authors are employed by the European Union, the copyright notice is: 978-1-5090-2807-8/16/$31.00©2016 European Union

The MMRE is defined as the average of all MRE’s (Magnitude of Relative Errors[3]).

#### image067 (16)

image069

Where, Ci is the real effort value of project i,

Ĉi is the predicted value of project i,

m represents a total number of projects.

PRED(0.25) is the percentage of estimations that fall within 25 percent of the original value[8].

image073 (17)

#### Considering the flaw in MRE that is it is biased towards underestimate predictions rather than the overestimate predictions, Miyazaki et al. (1990) proposed the MBRE (Mean Balanced Relative Error) metric. This metric overcomes this flaw by making the relative error unconfined to both overestimate and underestimate predictions. Therefore, the partiality of MRE metric is eliminated by taking the ratio of smallest value(between original and predicted values).

#### image077 (18)

#### image079

#### However, it has a limitation that is inevitably regarding linear regression models. Sometimes linear regression models have a tendency to produce negative predictions, so the results under MBRE appear bit distorted. Despite all these limitations of the metrics, MMRE has been believed as a good standard in the software development effort estimation literature.

# Experiments and Results

In this section, experimentation analysis can be conducted on three repository datasets, namely, Maxwell, Finnish, and Kemmerer, to assess the performance of our method.

## Dataset Preparation

Before going into the experiment, all the input variables (a. k. a features) undergo normalization so that the possibility of different influences can be eliminated [12]. The three datasets are presented in Table I are experienced to Leave-one-out cross-validation: all the data points in every iteration are used for training except the single observation, and the model is validated through that single observation [6]. In this experiment, the data points can be split into two sets: first, set contains two-third of the data points and performs the above-stated cross-validation, the second set contains one-third of the data points which is used for testing[7].Table I provides the descriptive statistics for the datasets above.

TABLE I. Descriptive Statistics For Public Datasets

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **Finnish** | **Kemerer** | **Maxwell** |
| Features | 8 | 7 | 27 |
| Size | 38 | 15 | 62 |
| Units | Hours | Months | Hours |
| Minimum effort | 460 | 26 | 583 |
| Median | 5430 | 1829 | 5189.5 |
| Mean | 7678.3 | 3921 | 8223.2 |
| Maximum effort | 26670 | 54620 | 63694 |
| Skew | 0.95 | 3.92 | 3.26 |

## Software Development Effort Estimation Models

The different methods used in this study are based on machine learning models and case-based reasoning approach. The machine learning models include LS-SVM: Least Squares Support Vector Machines, ELM: Extreme Learning Machine, ANN: Artificial Neural Network. The case-based reasoning model is ABE: Analogy-Based Software Effort Estimation which is hybridized with different machine learning models. The names of these hybridized methods are ABE-LS-SVM, ABE-ELM, ABE-ANN. The proposed estimation models, ABE-LS-SVM and ABE-ELM, in this study, are compared with the ABE-ANN effort estimation model to explore the flexibility of these models.

## Experimental Procedure

### The data points can be divided into training set and testing set in the experimental procedure. The training set and testing set includes two-third, and one-third of the data points respectively. For the purpose of validation, the training set carries out Leave-one-out cross-validation to get the measure of the generalization error of the methods[3,24,25,26]. In this procedure for each dataset of n training set data points, m testing-set data-points and given models, each model is trained with an n-1 training set data points and can be validated on the left-out sample[5]. The entire method repeats n times in anticipation of each training set data point in the dataset have been validated, and average training error can be computed. Now after the completion of the specified cross- validation, the testing set data points are exposed to the trained model to evaluate the models. The proposed model in this study have variants which play a role in estimating effort accurately. So, the optimal values of these variants are necessary. To obtain these values, by searching their solution space, optimization of these variants are achieved on the training dataset during the training phase. Now, the best variants of the proposed method can be obtained by summarizing and comparing the training and testing results[4]. The succeeding section presents the experimental results and analysis.

## Experimental Results

Table II displays the results generated by different methods across three different real-world datasets given in Table I. These results are expressed in the form of mean and standard deviations of above performance metrics. As this study is currently speaking about the ABE-LS-SVM, the behavior of the proposed method is potentially good when compared with the other two methods. By inspecting the results in Table II, the PRED (0.25) value of the proposed method is greater than the other two methods. The mean and standard deviations of MMRE, MBRE, and PRED interprets that the proposed method outperforms testing results of the ABE-ELM and ABE-ANN for Maxwell, Finnish and Kemmerer datasets.

# Conclusion

Analogy based estimation is a very primitive type of approach, and its empirical analysis has been done very extensively in software development effort estimation. In the current study, ABE is considered as a base method for the software development effort estimation. Since there is a need to adjust the retrieval solution, because the project being estimated and its analogy is not the same, the adjustment techniques enhance the retrieval solution to meet its project requirements. Also, as the importance of the precision of effort estimation is growing rapidly, exploring the several adjustment mechanisms also increasing. Until now, the exploration would be done mainly on linear adjustment models. Non-linear adjustment mechanisms, which have a learning ability to attune to any complex situations, are now gearing up to solve the uncertain relationship among the projects and confronting the non-normality and categorical features of different datasets. LS-SVM as a non-linear adjustment serves its best rather than ELM and ANN. As an extension to this study, there are other options for the kernel function in LS-SVM other than radial basis function.

TABLE II. Results Of All Datasets

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Dataset** | | **Maxwell** | | **Finnish** | | **Kemmerer** | |
| **Testing Error Metrics** | **Methods** | **Mean** | **Std.** | **Mean** | **Std.** | **Mean** | **Std.** |
| MMRE | ABE-LS-SVM | 1.1529 | 0.026257 | 1.7974 | 1.02 | 0.66412 | 1.31712e-09 |
| MBRE | 0.8106 | 0.024885 | 0.11798 | 0.029651 | 0.73495 | 6.7696e-10 |
| PRED | 0.42 | 0.021849 | 0.52 | 0.077327 | 0.40 | 5.6953e-17 |
| MMRE | ABE-ELM | 4.2891 | 1.183 | 2.3929 | 0.20714 | 1.8071 | 0.93002 |
| MBRE | 4.2104 | 0.19929 | 2.7883 | 0.16716 | -0.54375 | 1.7714 |
| PRED | 0.16 | 0.096977 | 0.15 | 0.017201 | 0.13 | 0.14903 |
| MMRE | ABE-ANN | 4.4466 | 1.5558 | 2.124 | 1.8193 | 2.0333 | 1.076 |
| MBRE | 3.4754 | 3.0643 | 0.38218 | 1.909 | 0.7181 | 3.3424 |
| PRED | 0.12 | 0.0889288 | 0.32 | 0.11049 | 0.08 | 0.13611 |

References

1. Li, Yan-Fu, Min Xie, and T. N. Goh. "A study of the non-linear adjustment for analogy based software cost estimation." *Empirical Software Engineering* 14, no. 6 (2009): 603-643.
2. Sun, Yongjun, Yiwei Liu, and Hong Liu. "Temperature Compensation for a Six-Axis Force/Torque Sensor Based on the Particle Swarm Optimization Least Square Support Vector Machine for Space Manipulator." *IEEE Sensors Journal* 16, no. 3 (2016): 798-805.
3. Benala, Tirimula Rao, Rajib Mall, Satchidananda Dehuri, and Pala Swetha. "Software Effort Estimation Using Functional Link Neural Networks Tuned with Active Learning and Optimized with Particle Swarm Optimization." In *International Conference on Swarm, Evolutionary, and Memetic Computing*, pp. 223-238. Springer International Publishing, 2014.
4. Benala, Tirimula Rao, Rajib Mall, and Satchidananda Dehuri. "Software Effort Estimation Using Functional Link Neural Networks Optimized by Improved Particle Swarm Optimization." In *International Conference on Swarm, Evolutionary, and Memetic Computing*, pp. 205-213. Springer International Publishing, 2013.
5. Laidi, M., C. Si-Moussa, and S. Hanini. "Novel approach for estimating solubility of solid drugs in supercritical carbon dioxide and critical properties using direct and inverse artificial neural network (ANN)." *Neural Computing and Applications*: 1-13
6. Li, Y. F., M. Xie, and T. N. Goh. "Bayesian Inference Approach for Probabilistic Analogy Based Software Maintenance Effort Estimation." In *Dependable Computing, 2008. PRDC'08. 14th IEEE Pacific Rim International Symposium on*, pp. 176-183. IEEE, 2008.
7. Li, Yan-Fu, Min Xie, and Thong Ngee Goh. "A study of project selection and feature weighting for analogy based software cost estimation." *Journal of Systems and Software* 82, no. 2 (2009): 241-252.
8. Chen, Guijun, Xueying Zhang, Zizhong John Wang, and Fenglian Li. "An Enhanced Artificial Bee Colony-Based Support Vector Machine for Image-Based Fault Detection." *Mathematical Problems in Engineering* 2015 (2015).
9. Li, Y. F., M. Xie, and T. N. Goh. "Optimization of feature weights and number of neighbors for analogy based cost estimation in software project management." In *2008 IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 1542-1546. IEEE, 2008.
10. Chou, Jui-Sheng, and Chieh Lin. "Predicting disputes in public-private partnership projects: Classification and ensemble models." *Journal of Computing in Civil Engineering* 27, no. 1 (2012): 51-60.
11. Li, Y. F., M. Xie, and T. N. Goh. "A study of mutual information based feature selection for case based reasoning in software cost estimation." *Expert Systems with Applications* 36, no. 3 (2009): 5921-5931.
12. Walkerden, Fiona, and Ross Jeffery. "An empirical study of analogy-based software effort estimation." *Empirical software engineering* 4, no. 2 (1999): 135-158.
13. Jørgensen, Magne, Ulf Indahl, and Dag Sjøberg. "Software effort estimation by analogy and “regression toward the mean”." *Journal of Systems and Software* 68, no. 3 (2003): 253-262.
14. Chiu, Nan-Hsing, and Sun-Jen Huang. "The adjusted analogy-based software effort estimation based on similarity distances." *Journal of Systems and Software* 80, no. 4 (2007): 628-640.
15. Li, Jingzhou, Guenther Ruhe, Ahmed Al-Emran, and Michael M. Richter. "A flexible method for software effort estimation by analogy." *Empirical Software Engineering* 12, no. 1 (2007): 65-106.
16. Li, Jingzhou, and Guenther Ruhe. "Analysis of attribute weighting heuristics for analogy-based software effort estimation method AQUA+." *Empirical Software Engineering* 13, no. 1 (2008): 63-96.
17. Angelis, Lefteris, and Ioannis Stamelos. "A simulation tool for efficient analogy based cost estimation." *Empirical software engineering* 5, no. 1 (2000): 35-68.
18. Pickard, Lesley, Barbara Kitchenham, and S. J. Linkman. "Using simulated data sets to compare data analysis techniques used for software cost modeling." *IEE Proceedings-Software* 148, no. 6 (2001): 165-174.
19. Sentas, Panagiotis, and Lefteris Angelis. "Categorical missing data imputation for software cost estimation by multinomial logistic regression."*Journal of Systems and Software* 79, no. 3 (2006): 404-414.
20. Liu, Qin, and Robert C. Mintram. "Preliminary data analysis methods in software estimation." *Software quality journal* 13, no. 1 (2005): 91-115.
21. Suykens, Johan AK, and Joos Vandewalle. "Least squares support vector machine classifiers." *Neural processing letters* 9, no. 3 (1999): 293-300.
22. Shepperd, Martin, and Chris Schofield. "Estimating software project effort using analogies." *IEEE Transactions on software engineering* 23, no. 11 (1997): 736-743.
23. Kolodner, Janet. *Case-based reasoning*. Morgan Kaufmann, 2014.
24. Benala, Tirimula Rao, Korada Chinnababu, Rajib Mall, and Satchidananda Dehuri. "A particle swarm optimized functional link artificial neural network (PSO-FLANN) in software cost estimation." In *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA)*, pp. 59-66. Springer Berlin Heidelberg, 2013.
25. Rao, B. Tirimula, Satchidananda Dehuri, and Rajib Mall. "Functional link artificial neural networks for software cost estimation." *International Journal of Applied Evolutionary Computation (IJAEC)* 3, no. 2 (2012): 62-82.
26. Benala, Tirimula Rao, Satchidananda Dehuri, and Rajib Mall. "Computational intelligence in software cost estimation: an emerging paradigm." *ACM SIGSOFT Software Engineering Notes* 37, no. 3 (2012): 1-7.
27. Huang, Xiaolin, Andreas Maier, Joachim Hornegger, and Johan AK Suykens. "Indefinite kernels in least squares support vector machines and principal component analysis." *Applied and Computational Harmonic Analysis* (2016).