A Software Measurement Using Artificial Neural Network and Support Vector Machine

Bhavendra Kumar Sinha[1], Amit Sinhal[2] and Bhupendra Verma[3]

[1]M.Tech. (IT) Student, Department of Information Technology, TIT, Bhopal
Bhavendra_sinha@yahoo.com
[2]Professor, Department of Computer Science & Engineering, TIT, Bhopal
amit_sinhal@rediffmail.com
[3] Director, TIT (Excellence), Bhopal
bk_verma3@rediffmail.com

ABSTRACT

Today, Software measurement are based on various techniques such that neural network, Genetic algorithm, Fuzzy Logic etc. This study involves the efficiency of applying support vector machine using Gaussian Radial Basis kernel function to software measurement problem to increase the performance and accuracy. Support vector machines (SVM) are innovative approach to constructing learning machines that Minimize generalization error. There is a close relationship between SVMs and the Radial Basis Function (RBF) classifiers. Both have found numerous applications such as in optical character recognition, object detection, face verification, text categorization, and so on. The result demonstrated that the accuracy and generalization performance of SVM Gaussian Radial Basis kernel function is better than RBFN. We also examine and summarize the several superior points of the SVM compared with RBFN.

KEYWORDS

Support Vector Machine (SVM), Radial Basis Function (RBF) Network, Software Measurement.

1. Introduction

Software Measurement is the process which helps the organizations to improve a measurement program. It does not provide guidance for application of some measurement such as software cost measurement or to analyze software complexity. Instead of it does help on effective software measurement program and for understanding some of the key lessons that how the work. Klayman et al. (1999) reported that the measurement is accurate judgment process which depends upon confidence people but overconfidence increases with the difficulty of the task [1]. The purpose of the Software Measurement has grown up of successful measurement applications. Stone and Opel (2000) report about probability judgment accuracy by examining the effects of two different training techniques such as performance feedback or environmental feedback. They measured their improvement in calibration and discrimination as a function of feedback type which require separate training techniques for improvement [2]. It presents the specific procedures and activities which play the roles to achieve the goals and also roles of the people involved.

Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to characterize the attributes by clearly defined rules. Thus, measurement requires

- Entities (objects of interest).
- Attributes (characteristics of entities).
- Rules (and scales) for assigning values to the attributes.

Entity is an object or an event in the real world. An entity is important to distinguish one entity from another by identifying characteristics.

An attribute is a feature or property of an entity. An entity defined by attributes and attribute using numbers or symbols. These numbers and symbols are abstraction that we use to reflect our perceptions of the real world. Software entities are described by attributes such as size, cost, elapsed time, effort expended, response time, transaction rates, number of defects found, and operational reliability.

In software measurement various factors have made development effort and cost difficult, complex and inaccurate. These factors influencing total development time between expert programmer and average programmer team. Software development cost measurement models can be used to determine development effort which can then transformed into the development time and cost. Although, these all are closely related with each others.

Accurate Software measurement is important for reasons such as

- It can provide the facility for classify and design different development projects for various business applications.
- The measurement model can determine how the resources can be used in well manner.
- It can be used to assess the impact of changes and support preplanning.
- The resources can be used to fulfill the needs of project application.
- Customers expect actual development costs and time to be in line with estimated costs and time.

2. A Review on Software Measurement

This software measurement based survey presents information to show the importance of measurement. It discusses significant specific strategic procedure and activity for software project measurement in terms of quality, complexity, cost, effort and schedule estimation. Software cost is related closely to software quality and productivity. Unrealistically low cost estimates frequently lead to poor product quality and low project productivity [3].

- G. R. Finnie et al. (1997) conclude that an artificial intelligence models are capable of providing adequate measurement models. Their performance is to a large degree dependent on the data on which they are trained, and the extent to which suitable project data is available will determine the extent to which adequate effort measurement models can be developed [4].
- Anita Lee et al. (1998) concluded that an integrated neural network model with cluster analysis is provide accurate measurement for software cost measurement to increase the training efficacy of the network [5].
- Hearst et al. (1998) positioned the SVM algorithm at the intersection of learning theory and practice: ‘‘it contains a large class of neural nets, radial basis function (RBF) nets, and polynomial classifiers as special cases. Yet it is simple enough to be analyzed mathematically, because it can be shown to correspond to a linear method in a high dimensional feature space nonlinearly related to input space.’’ [6].

- Tony Van Gestel et. al. (2001) was described in this paper; the Bayesian evidence framework is combined with least squares support vector machines (LS-SVMs) for nonlinear regression in order to infer nonlinear models of a time series and the corresponding volatility. On the first level of inference, a statistical framework is related to the LS-SVM formulation which allows to include the time-varying volatility of the market by an appropriate choice of several hyperparameters. In the second level inferred hyperparameters, related to the volatility, are used to construct a volatility model within the evidence framework. Model comparison is performed on the third level to infer the tuning parameter of the RBF-kernel by ranking the evidences of the different models [7].
- Zan Huang et. al. (2004) reported in their study a newly introduced learning method based on statistical learning theory, support vector machines, together with a frequently used high-performance method, backpropagation neural networks, to the problem of credit rating prediction. They used two data sets for Taiwan financial institutes and United States commercial banks for experiment. The results showed that support vector machines achieved accuracy comparable to that of backpropagation neural networks [8].
- M. Jorgensen (2004) concludes on their review that expert estimation is the dominant strategy when estimating the effort of software development projects, and that there is no substantial evidence supporting the superiority of model estimates over expert estimates. There are situations where expert estimates are more likely to be more accurate, e.g., situations where experts have important domain knowledge not included in the models or situations when simple estimation strategies provide accurate estimates [9].
- Mohammed Abdullah Al-Hajri et. al. (2005) report that in software measurement includes the standard Function Point (FP) and many different models derived from it. In this paper a new FP weights system was established using Artificial Neural Networks. This method is a modification of the complexity weights of FP measure and the results were very accurate and much suitable when they were applied on real data sets of software projects [10].
- Kai-Bo Duan and S. Sathiya Keerthi (2005) stated that Multiclass SVMs are usually implemented by combining several Binary (two-class) support vector machines (SVMs) classification techniques which is a very well developed technique. Due to various complexities, a direct solution of multiclass problems using a single SVM formulation is usually avoided. The better approach is to use a combination of several binary SVM classifiers to solve a given multiclass problem. Popular methods are (WTA_SVM); (MWV_SVM). In the paper numerically study the performance of the four methods such as winner-takes-all strategy (WTA_SVM), max-wins voting (MWV_SVM), pairwise coupling with Platt’s probabilities (PWC_PSVM) and pairwise coupling with Kernel logistic regression (PWC_KLR). PWC PSVM gives the best classification results and has significantly smaller mean values of test error but in WTA SVM, MWV SVM and PWC KLR, it is hard to tell which one is better [11].
- Yuming Zhou et. al. (2007) report Accurate software metrics-based maintainability prediction can not only enable developers to better identify the determinants of software quality and thus help them improve design or coding, it can also provide managers with useful information to help them plan the use of valuable resources. In this paper, The prediction accuracy of the MARS models are evaluated and compared using multivariate linear regression models, artificial neural network models, regression tree models, and support vector models. The results suggest that for one system MARS can predict maintainability more accurately than the other four typical modeling techniques [12]
- YuShen Su and Chin-Yu Huang (2007) propose an artificial neural-network-based approach for software reliability measurement based on numerical analysis viewpoints of software reliability modeling and applying this approach to build a dynamic weighted combinational model (DWCM). In their experimental analysis two real software failure

data sets are analyzed by which gives you a better predictions result for proposed dynamic weighted combinational mode [13].

- Dr. Karanjeet Singh Kahlon et al. (2010) propose of measurement by selecting the most popular models among Artificial-Neural-Network Based Model (ANN) and Halstead, Walston-Felix, Bailey-Basili and Doty models to improve accuracy to utilize NASA software projects. The proposed Neuro based system is able to provide good measurement capabilities [14].
- Chintala Abhishek et al.(2010) focuses on finding a method which gives a best analysis report for effort prediction in testing phase by using large number of test cases by selecting the unique feature of learning through usage. The proposed model developed can also be evolved a minor effect on the effort measurement which affect the accuracy of model [15].

3. ANN Overview

The first neural network architecture that we have chosen is the Ward Network [16]. Neural network is a highly interconnecting network of a complex and large number of processing element called neurons. Neural network architecture classify besides processing on input and output layer and also have one or more intermediately layer called hidden layer. The input layer neurons are linked to hidden layer neurons and the hidden layer neurons are linked to output layer called Hidden output layer weights. An artificial neural network (ANN) is an efficient information processing system which resembles in characteristics with biological neural networks. ANN process large no of highly interconnected processing element called nodes or units or neurons, which usually operate in the parallel and are configured in regular architectures shown in Figure 1. Each neuron is connected with the other by a connection link. Each connection link is associated with weights which contain information about input signal. This information is used by the neurons net to solve a particular problem.

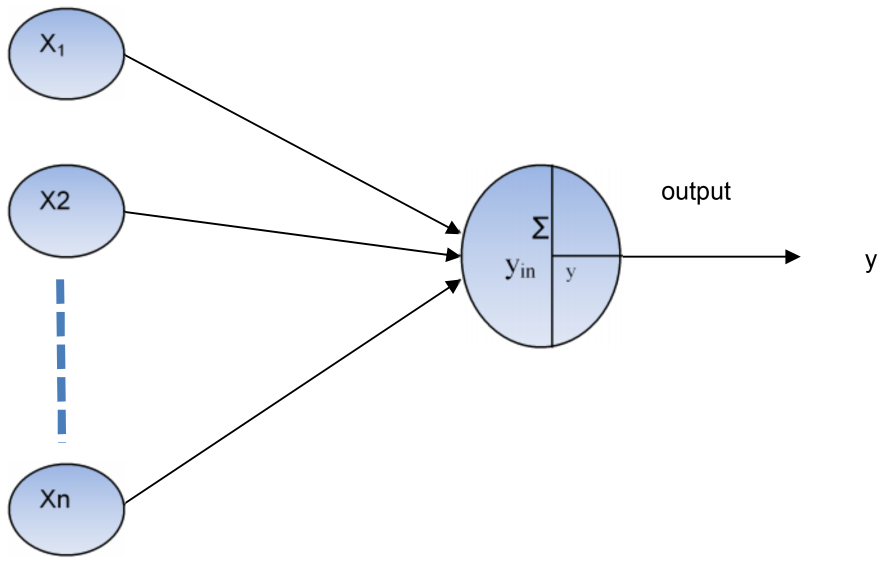

Figure 1:- Mathematical model of artificial neuron

4. Neural Network Learning Algorithm Classification

The classification of the neural network learning algorithm is illustrated as in Figure 2;-

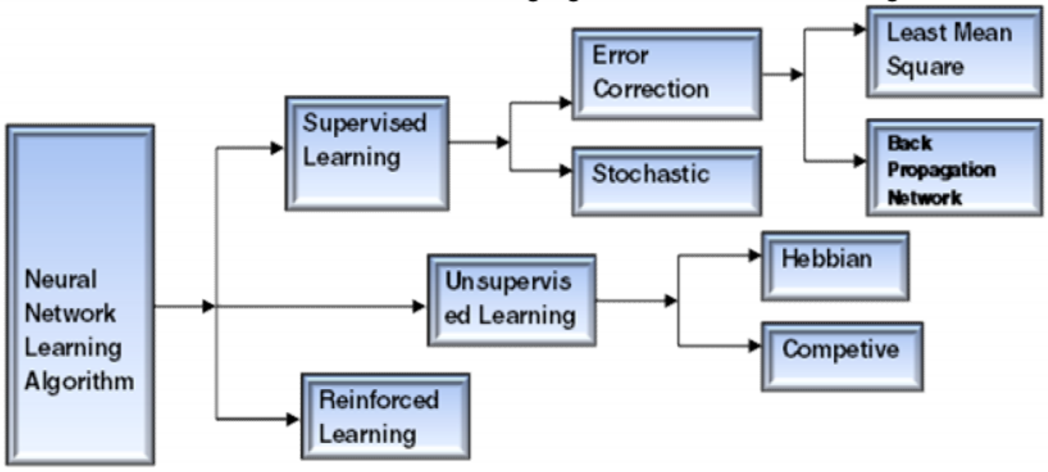

Figure 2: Neural Network Learning Algorithm

The neural network learning algorithms have been categorized as Supervised Learning (SL), Unsupervised Learning (USL) and Reinforcement Learning.

In Supervised Learning every input pattern is used to train the network is associated with the output patterns. During the Supervised Learning process the computation are based on actual and expected output to determine the errors and the errors can be used to improve the performance of the network model. In the Unsupervised Learning method the target output is not given to the network model. Now the system will capable to learn of its own desired pattern and observing expected structure for input pattern. In Reinforce Learning (RL) algorithm is used both SL and USL algorithm approach for prediction of result where the various options related with correct option is given to the examiner and the examiner would capable to identify the correct expected answer by using their own approaches.

4.1 Radial Basis Function Network

RBF networks have been proven to be universal approximates and have the properties of fast convergence, easy solution to regularization, and robustness to outliers [17]. The construction of Radial Basis Function network (RBFN) based on three layer architecture namely, an input layer is made up of source nodes that connected to the network. The second layer hidden layer with non-linear Radial Basis activation function, such as Gaussian function that receives non linear transformation from the input layer to the hidden layer. The output layer implements a linear combination of Radial Basis Function that constitutes hidden layer function in a network to the activation pattern applied on it. Miyoung Shin (2000) pointed out that many desirable features and properties of RBF are flexible and potentially very useful for software measurement application [18]. The RBFN is used to applied on to focus complex pattern classification problem is basically solved by the transforming it into a high dimensional space in nonlinear manner.

Step 1:- Set the weight to small random values.

Step 2:-Perform Step 2-8 when the stopping condition is false.

Step 3:- Perform Step 4-8 for each input.

Step 4:- Each input unit receives input signals and transmits to the next hidden layer unit.

Step 5:- Calculate the radial basis function.

Step 6:- Select the centers for the radial basis function. The centers are selected from the Set of input vectors. It should be noted that a sufficient number of centers have to be selected to ensure adequate sampling of the input vector space.

Step 7:- Calculate the output from the hidden layer unit:

$$V_i(x_i) = \frac{\exp[-\sum (x_{ij} - \hat{x}_{ij})^2]}{\sigma_i^2} \qquad (1)$$

where

$\hat{x}_{ij}$ = Center of the RBF unit for input variables

σ_i = width of ith RBF unit

x_{ji} = jth variable of input pattern

Step 8 :- Calculate the output of the neural network :

$$\gamma_{net} = \sum_{i=1}^{k} w_{im} V_i(x_i) + w_o \qquad (2)$$

where

k = number of hidden layer nodes (RBF function).

ϒnet = output value of mth node in output layer for the nth incoming pattern.

w_{im} = weight between ith RBF unit and mth output node.

w_o = biasing term at nth output node.

Step 9: Calculate the error and test for the stopping condition. The stopping condition may be number of epochs or to a certain extent weight change

.

5. Radial Basis Function Optimization using Support Vector Machine

Support vector machines (SVM) are innovative approach to constructing learning machines that Minimize generalization error. SVMs are based on very simple and intuitive concepts. They are constructed by locating a set of (hyper) planes that separate two or more classes of data. By construction of these hyper planes, the SVM discovers the boundaries between the input classes; the elements of the input data that define these boundaries called support vectors. SVMs can classify data separated by nonlinear boundaries. Through the use of kernel function, it problem is implicitly

mapped to a higher dimensional space in which hyper planes suffice to define the boundaries.
Let us define labeled training examples [vi, yi], an input vector $v_i \epsilon R^n$, a class value y_i $\epsilon\{-1,1\}$, i=1,.,ι. Here following SVM algorithm define hyper plane decision function for non-linearly separable case and multi class support vector classifier.

Step 1 :- Initialize Data needs to be trained under normal and faulty condition.
Step 2 :- Build the structure of SVM and Define the optimal hyper planes in terms of the support vector

$$Y_{out} = sign(\sum_{i=1}^{t} y_i \propto_i .(v.v_i) + b) \quad (3)$$

Where Y_{out} is the outcome, y_i is the class value of the training example vi, and . represents the inner product. The vector v= $(v_1,v_2,.,v_t)$ corresponds to an input and the vectors v_i , i=1,.,t, are the support vectors. In Eq. (3), b and α_i are parameters that determine the hyper plane.

Step 3 :- For the non-linearly separable case, a high-dimensional Version of Eq. (3) is given as follows:

$$Y_{out} = sign(\sum_{i=1}^{t} y_i \propto_i .k(v,v_i) + b) \quad (4)$$

The function $k(v,v_i)$is defined as the kernel function and formula for a Gaussian radial basis function machine with kernel function is given a
s

$$k(v,v_i) = (-\gamma|v - v_i|^2) \quad (5)$$

Where, γ is radial basis function kernel.
Step 4:- Determine the constraint C by cross validation.
Step 5:- Construct the classification function and Compute the SVM model.
Step 6:- Remove low rank feature and stop when features will be ended.

6. RESULT SIMULATION

The present work have tried to find out the Development Time (DT) by applying first the Neural Radial Bias Function Network Model and then the Support Vector Machine. Following methodology was adopted to carry out the development time prediction measurement using both the approaches. The evolution criteria of performance analysis is based on Magnitude of Relative Error (MRE), Mean Magnitude of Relative Error (MMRE) and Percentage of the Prediction (PRED) which are mentioned below as .:-

MRE = |ADT-EDT|/ADT

$$MMRE = \sum_{i=1}^{N} MRE/N$$

PRED(X) = P/N

Where, **ADT** is actual development time and **EDT** is estimated development time and **P** is the number of instances of projects with MRE less than or equal to **X** which is accepted at level of 0.25 and N is the number of considered projects used. The selected model comparisons are given in Table 1 with various criteria such as Standard Deviation (SD), Mean Absolute Error (MAE) and with Min, Max and Mean error values.

Table 1:-Experimental Result Evaluation of Proposed Model

Measurement Model	**Mean**	**SD**	**Min**	**Max**	**MAE**
RBFN	0.619	2.83	-5.639	6.59	2.09
SVM	0.299	2.194	-3.72	5.055	1.772

Table 2 shows the dataset used for carrying out experimentation which has Lopez-Martin dataset with results of MRE and MMRE of both model RBFN and SVM with Gaussian Radial basis kernel function.

Table 2 : Detailed Lopez-Martin dataset with their standard numeric values.

Lopez-Martin Data					**MRE (Mean Relative Error)**	
S No	**mc**	**dc**	**loc**	**dt**	**RBFN**	**SVM**
1	1	0.25	4	13	0.476545	0.059047
2	1	0.25	10	13	0.312803	0.025356
3	1	0.333	4	9	0.593678	0.193875
4	2	0.083	10	15	0.129818	0.135453
5	2	0.111	23	15	0.049379	0.170883
6	2	0.125	9	15	0.038338	0.073344
7	2	0.125	9	16	0.098442	0.00626
8	2	0.125	14	16	0.034378	0.032723
9	2	0.167	7	16	0.100582	0.057817
10	2	0.167	8	18	0.126528	0.157953
11	2	0.167	10	15	0.168749	0.021511
12	2	0.167	10	15	0.168749	0.021511
13	2	0.167	10	18	0.026043	0.148741
14	2	0.2	10	13	0.079027	0.126504
15	2	0.2	10	14	0.001954	0.046039
16	2	0.2	10	15	0.064843	0.023697
17	2	0.2	15	13	0.189461	0.158854

18	2	0.25	10	12	0.021553	0.135947
19	2	0.25	10	12	0.021553	0.135947
20	3	0.083	17	22	0.235077	0.150225
21	3	0.125	11	19	0.016144	0.087879
22	3	0.125	15	18	0.048932	0.017935
23	3	0.125	15	19	0.006275	0.069622
24	3	0.143	13	21	0.005145	0.184149
25	3	0.143	14	20	0.070109	0.139021
26	3	0.143	14	21	0.019152	0.18002
27	3	0.143	15	19	0.036894	0.08915
28	3	0.143	15	20	0.01495	0.134693
29	3	0.167	13	15	0.234724	0.108913
30	3	0.167	14	13	0.433771	0.286179
31	3	0.2	18	19	0.016411	0.13808
32	3	0.25	9	13	0.035132	0.120768
33	3	0.25	12	12	0.246924	0.235027
34	3	0.25	17	12	0.337953	0.27072
35	4	0.077	16	21	0.184967	0.058482
36	4	0.077	31	21	0.228694	0.004777
37	4	0.111	16	19	0.056199	0.005272
38	4	0.2	24	18	0.058955	0.003508
39	5	0.143	22	24	0.232901	0.168949
40	5	0.143	22	25	0.263585	0.202191
41	5	0.2	22	18	0.052967	0.04327
MMRE (Mean Magnitude Relative Error)					**0.13508**	**0.108056**

In the Table 3 MMRE and PRED (0.25) criteria is analyzed over the complete dataset. It can be seen that the SVM model outperform the RBFN model. The performance of SVM and RBFN model is compared for development time prediction problem in the present work.

Table 3: Result analysis of different prediction model with various performance criteria

Performance Criteria	**Prediction Model**	
	RBFN	**SVM**
MMRE	0.13508	0.108056

PRED(0.25)	0.75609	0.951219

The results have minimized error as compared with actual value. These results emphasize the importance of software development time measurement. Figure 3 shows the behavior of measurement of the optimal software development time in available dataset used in thesis work. In the Figure 3 red fluctuated line denotes the estimated development time mean relative error (MRE) of SVM model and compares it with blue fluctuated line of RBFN model. From the Figure 3, it is found that both RBFN and

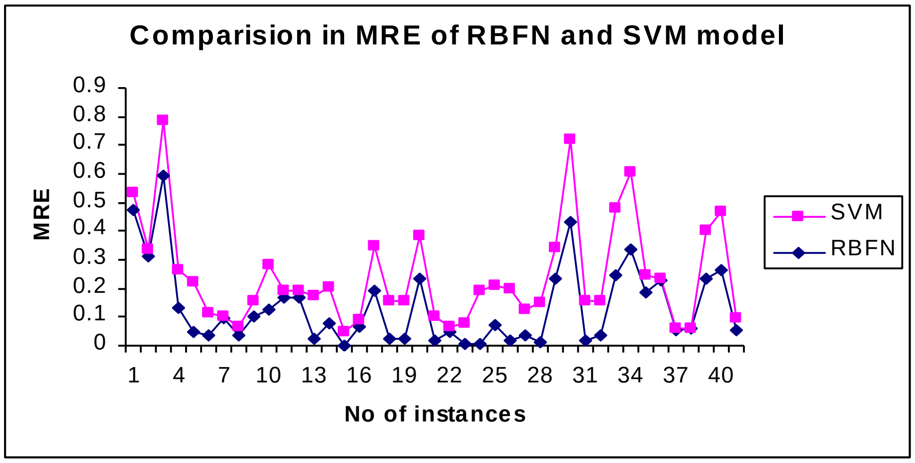

Figure 3: Comparison in MRE between both Software Measurement model RBFN and SVM

SVM model fluctuated in the neighborhood of the actual development time and MRE is taken on various observation points.

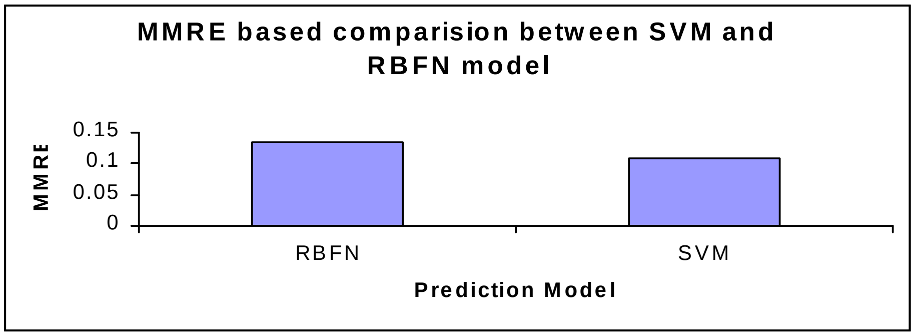

Figure 4: Graph on MMRE (Mean Magnitude Relative Error) of different Prediction Model.

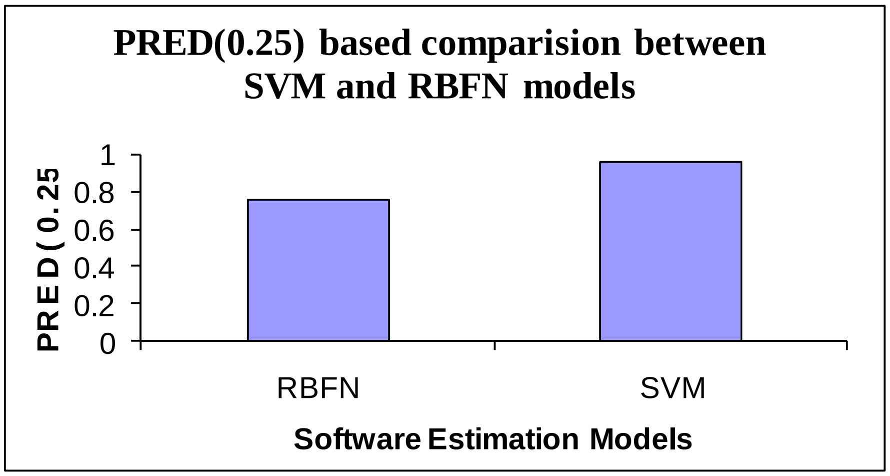

Figure 5: Graph on PRED (0.25) of different Software Prediction Model.

From the graph drown in Figure 4 and Figure 5 based on result analyzed by Table-3, it is clearly recognize that SVM approach give better performance then the existing RBN model.

5. CONCLUSION

This work illustrated a review on neural network and SVM approaches for software measurement to predict software development time. The proposed analysis gives the better result to compare their work done and result over the year. The review clearly points out the potential of artificial neural networks being used as a tool for classification and prediction problems. This work evaluates the capability of SVM using Gaussian basis kernel function which have been successfully applied for solving both classification and regression problems in many applications. The results showed that support vector machines achieved accuracy comparable to that of redial bias neural networks. The methodology can be used to helpful for proposing an efficient method using both techniques such as RBFN and SVM which can be applied over the other measurement model for software measurement with the efficient manner.

REFERENCES

[1] Klayman, J., Soll, J.B., Gonzalez, V.C., Barlas, S., (1999) "Overconfidence: it depends on how, what and whom you ask", Organizational Behaviour and Human Decision, Vol. 79, No. 3, pp. 216–247.
[2] Stone, E.R., Opel, R.B., (2000) "Training to improve calibration and discrimination: the effects of performance and environmental feedback", Organizational Behavior and Human Decision Processes, Vol. 83, No. 2, pp. 282-309.
[3] W. E. Lehder, D. P. Smith, and W. D. Yu, (1998), "Software Estimation Technology, "AT&T Tech. J., Vol. 67, No. 1, pp. 10-18.
[6] M.A. Hearst, S.T. Dumais, E. Osman, J. Platt, B. Scholkopf, (1998) "Support vector machines", IEEE Intelligent Systems, Vol. 13, No. 4, pp. 18– 28.
[5] A. Lee, C.H Cheng, J. Balakrishnan (1998) "Software development cost estimation: Integrating neural network with cluster analysis", Elsevier Science B.V., Vol. 34, No. 1, pp. 1-9.
[6] G. R. Finnie, G. E., Wittig and J-M. Desharnais, (1997) "A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models", Journal of System and Software, Vol. 39, No. 3, pp. 281-289.
[7] Tony Van Gestel, Johan A. K. Suykens, Dirk-Emma Baestaens, Annemie Lambrechts, Gert Lanckriet, Bruno Vandaele, Bart De Moor, and Joos Vandewalle, (2001)" Financial Time Series

Prediction Using Least Squares Support Vector Machines Within the Evidence Framework", IEEE Transactions on Neural Networks, Vol. 12, No. 4, pp. 809 - 821
[8] Zan Huanga, Hsinchun Chena, Chia-Jung Hsua, Wun-Hwa Chenb, Soushan Wu, (2004) "Credit rating analysis with support vector machines and neural networks: a market comparative study", Decision Support Systems, Vol. 37, No. 4, pp. 543–558.
[9] M. Jorgensen, (2004) "A review of studies on expert estimation of software development effort",The Journal of Systems and Software, Vol. 70, No. 1-2, pp. 37–60.
[10] Mohammed Abdullah Al-Hajri , Abdul Azim Abdul Ghani, Md Nasir Sulaiman, Mohd Hasan Selamat, (2005) "Modification of standard Function Point complexity weights system", The Journal of Systems and Software, Vol. 74, No. 2, pp. 195-206.
[11] Kai-Bo Duan and S. Sathiya Keerthi, (2005) "Which Is the Best Multiclass SVM Method? An Empirical Study", Springer Berlin Heidelberg, Vol. 3541, pp. 278-285.
[12] Yuming Zhou, Hareton Leung, (2007) "Predicting object-oriented software maintainability using multivariate adaptive regression splines", The Journal of Systems and Software, Vol. 80, No. 8, pp. 1349–1361.
[13] Yu-Shen Su, Chin-Yu Huang, (2007) "Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models", The Journal of Systems and Software, Vol 80, No. 4, pp. 606-615.
[14] Jaswinder Kaur, Satwinder Singh, Dr. Karanjeet Singh Kahlon, Pourush Bassi, (2010) " Neural Network-A Novel Technique for Software Effort Estimation ",International Journal of Computer Theory and Engineering, Vol. 2. No. 1, pp. 1793 – 8201.
[15] Chintala Abhishek, Veginati Pavan Kumar, Harish Vitta, Praveen Ranjan Srivastava (2010) "Test Effort Estimation Using Neural Network", Journal of Software Engineering & Applications (JSEA), Vol. 3, pp. 331-340.
[16] NeuroShell 2 Help, Ward Systems Group Inc., http://www.wardsystems.com
[17] A. K. Jain, R. P. W. Duin and J. Mao, (2000) "Statistical Pattern Recognition: A Review," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 1, pp. 4-37.
[18] Miyoung Shin and Amrit L. Goel, (2000) "Empirical Data Modeling in Software Engineering Using Radial Basis Functions", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Vol 26, No. 6, pp. 567 – 576.

AUTHORS

Bhavendra Kumar Sinha is a M. Tech student in the Technocrats Institute of Technology, Bhopal (MP). He received Bachelor of engineering degree in Information Technology from GuruGhasiDas Vishwavidyalya, Bilaspur(CG). His main research interest include software metrics and measurement, software engineering application and application of softcomputing and data mining in software engineering.

Prof. Amit Sinhal completed his B.E. in Computer Engineering from NIT Surat in 1996, M.Tech in Computer Science & Engineering from SATI Vidisha in 2005 and is pursuing Ph.D. from Rajiv Gandhi Technical University, Bhopal. He worked in various reputed software development companies as Project Lead and University Institute of Technology, Barkatullah University Bhopal as Assistant Professor. Currently he is working at Technocrats Institute of Technology, Bhopal as Professor in the Computer Science & Engineering department. He has published/presented more than 20 research papers in International Journals/Conferences.

Dr. Bhupendra Verma completed his B.E. in Computer Engineering, M.Tech in Computer Science & Engineering from SATI Vidisha and Ph.D. from Rajiv Gandhi Technical University, Bhopal. Currently he is working in Technocrats Institute of Technology, (Excellence) Bhopal as Director and Professor in the Computer Science & Engineering department. He has published/presented more than 50 research papers in International Journals/Conferences. He has guided 20 M.Tech dissertations and is presently guiding five students for Ph.D.